Volume 27 Number 4 December 2003 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

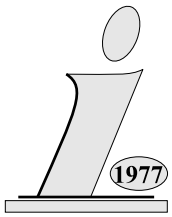
Special Issue:

Information Society

Guest Editors:

Marcin Paprzycki

Matjaz Gams



The Slovene Society Informatika, Ljubljana, Slovenia

EDITORIAL BOARDS, PUBLISHING COUNCIL

Informatica is a journal primarily covering the European computer science and informatics community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the list of referees. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Introduction:

Information Society; Foreword to the Special Issue

In recent years, due to its expected profound consequences on everyday lives, *Information Society* has become a hot topic of discussions among non-specialists as well as one of the important directions of development and research in computer science and informatics. The use of mobile devices and the Internet has become a standard even for those living in less developed countries, and has laid the technological foundation for the Information Society. It is often claimed that the Information Society has a potential to change our lives not only through the introduction of new information services but also by significantly affecting social relations. The interplay of complex forces involved in the development of the multiple facets of the Information Society shapes the problem space for research and necessitates interdisciplinary approaches to finding solutions. In response to these challenges, the Information Society Multiconference was created, whose goal is to become a place for exchange of knowledge related to information, communication and computer services, a place where researchers can present, analyze, and verify new discoveries in the scientific community first thus preparing a ground for their enrichment and development in practice. The Multiconference utilizes its Central-European location to became a bridge, through which the Western European experiences are transferred to Central and Eastern Europe, while enriching Western thought by the unique experiences brought about by the transformations happening in Eastern and Central Europe. It is supported and co-organized by several major research institutions and societies. In 2003, we proudly announce cooperation with ACM Slovenia, i.e. the Slovenian chapter of the ACM, the largest worldwide society in computer science and informatics. We would also like to use this opportunity to thank the Slovenian government for cooperation and support, in particular through the Ministry of Education, Science and Sport, and the Ministry of Information Society.

The Information Society special issue of the Informatica Journal consists of selected extended versions of papers presented at the 6th annual Information Society Multiconference. Each year, the Multiconference consists of 6 to 7 conferences and consists of over 200 paper presentations, round-tables, and discussions (many of these events are recorded and can be viewed through the Internet at http://ai.ijs.si). Typically, Multiconference contributions are published in 2 to 4 separate proceedings volumes totaling 500 to 700 pages and selected papers become published special journal issues, and this issue of Informatica is one of them.

The 6th Information Society 2003 Multiconference consisted of the following conferences:

- Collaboration and Information Society
- Complex Systems in E-Business CSeB'03
- Development and Reengineering of Information Systems
- Education in Information Society
- Intelligent and Computer Systems
- Management and Information Society
- Parallel and Distributed Computing
- Theoretical Computer Science
- Cognitive Sciences

Out of them we have selected the following papers to be extended and published in this Special Issue:

"*Trust and Fraud on the Internet*" by Bezalel Gavish deals with one of the essential problems on the Internet. Fraud can be a major stumbling block for increasing the volume of electronic commerce on the web, thus influencing research on trust building activities given the prevalence of fraudulent activities on the web.

"Learning Customer Profiles Using Unlabelled Data" by Giovanni Semeraro, Pasquale Lops and Marco Degemmis presents a recommender system that exploits supervised learning methods to learn user profiles from items previously rated by users. Profiles are used to find, classify, or rank items that are likely to be of interest to the user.

"Synergetic Integration of Aglets and E-speak in E-Commerce" by Andreas Schmid and Tong-Seng Quah proposes and implements a system for seamless integration of HP E-speak E-services and IBM Aglet mobile agent platform into a common platform. The developed software enables all applications written in Aglets and E-speak to create joint services across the two platforms. It establishes an environment for collaboration, intercommunication, translation, and bridging.

"*An Impact of ICT – Assessment of Indicators on National and Companies' Level***"** by Cene Bave, Maja Bučar and Metka Stare presents a discussion and assessment of statistical indicators used at different levels of decision making – from national to companies' levels. The case of Slovenia is presented to illustrate interpretations and assess presently used indicators.

"Categorization of Numerical Values for DEX Hierarchical Models" by Martin Znidarsic, Marko Bohanec and Ivan Bratko describes DEX - a multi-attribute decision modeling methodology with emphasis on numerical data categorization. Two methods suggested in the paper are: interval bounds according to the desired number of categories and the preference curve of attribute values.

"Evolutionary Optimization of Markers in Clothes Production" by Bogdan Filipic, Iztok Fister and Marjan Mernik describes optimization of markers in preparation of industrial production of clothes. The evolutionary algorithm is used to optimize the NP-hard problem. Measurements show superior performance compared to other tested algorithms.

"Increasing Fault-Tolerance of Multi-Agent Systems" by Andraz Bezek and Matjaz Gams analyzes and proposes a new approach to the problem of fault tolerance in load balancing systems. The presented ideas are applied in a multi-agent system for fault-tolerant network load balancing showing their advantage in specific situations.

"*Empirical Assessment of Methods for Software Size Estimation*" by Aleš Živkovič, Marjan Heričko and Tomaž Kralj present several methods for software size estimation such as the Function Points Analysis (FPA) method. A mathematical model is defined and used for theoretical comparison. Empirical results showed some limitations of the mapping function and anomalies in the data set used.

Marcin Paprzycki and Matjaž Gams

Trust and Fraud on the Internet

Bezalel Gavish
Cox School of Business, Southern Methodist University, Dallas, TX 75205,
e-mail: gavishb@charter.net

Keywords: Electronic commerce, Fraud, Trust

Received: June 3, 2003

Trust is an important ingredient for fostering the development of electronic commerce on the Internet. Recent surveys report on the growth of fraudulent activities on the Internet. We report the results of such an investigation which checked for the amount of fraudulent activities on auction sites. The result shows a significant level of such activities, well above the levels reported by auction site operators. Since fraud can be a major stumbling block for increasing the volume of electronic commerce on the web, it motivated research on trust building activities given the prevalence of fraudulent activities on the web.

1 Introduction

Mutual trust is recognized as an important factor in facilitating further development of electronic commerce on the Internet. Widespread electronic trades and exchanges can take place if the traders can trust each other and the procedures used to conduct the trade. A significant number of investigators have addressed trust issues, however due to its complex nature the formation and management of trust in web based environments raises many new research issues. One of the main concerns in trading on the internet is the virtual interaction between entities whose physical attributes are not known and can be misrepresented. By its anonymous nature, virtual interaction is a fertile ground for fraudulent activities [1-3].

Understanding the mechanisms used by swindlers in their fraudulent activities, and the resulting level of fraud, is especially important due to the "network externality" property. The network externality property is one in which a large number of traders (buyers and sellers) attracts other potential traders to join the trading process, this leads to a larger number of traders, this effect is based on the knowledge that satisfied traders induces others to trade on the Internet increasing the liquidity and efficiency of trades. If more than a handful of traders perceive that the system does not work in a fair and beneficial manner, it will cause a reverse process in which the number of traders will be significantly reduced, leading to further reduction in trading activity. This positive feedback vicious cycle can significantly reduce the potential for economic benefits of electronic commerce.

Fraud on the Internet is developing into a major issue of concern for consumers and businesses. The Financial Times reports that online fraud represents "an epidemic of huge and rapidly growing proportions" and that the incidence of fraud is 20 times higher in online trades than in offline trades (Waters [9]). VISA claims that half its credit card fraud complaints come from Internet based transactions, this in-spite of the fact that only 2 percent of its transactions originated on the Internet (BBC [1]), According to VeriSign study [8] 6.25% of e-commerce transactions carried out in the U.S. were attempts at fraud. More than half the fraud attempts were made by entities outside the U.S., Also, the number of security fraud incidents almost doubled between May 2003 and August 2003. One area that is of particular interest is the one of fraud in internet based auctions. The number of cases of auction fraud reported to the Federal Trade Commission in the USA has jumped from 106 in 1997 to around 25,000 in 2001 (Reuters, [6]). At the same time major Internet auction sites estimate that fraudulent outcomes occur only once in approximately 10,000 auctions, which seems at odds with the above trends and statistics.

Assuming that auction site claims of such a low level of fraudulent activity is true, trading activity can be encouraged by charging a minimal premium on each transaction. The collected premiums can provide automatic insurance against fraud to the trading parties. If the above numbers are correct, a surcharge of one cent per ten dollars value of trade should suffice to provide trade insurance. That level of premiums is a minuscule expense to the trading parties and leaves plenty of additional profit to the insuring entity. The fact that such insurance is not offered at this level of premium casts doubt about the validity of claims of such low levels of fraudulent activity[1].

Providing low cost insurance is not as simple as it sounds. Having low cost automatic insurance can have an effect on the behaviour of the trading parties. One possible outcome could be their tolerance of higher levels of risk in their trading activity, this can lead to

[1] Buyer insurance is offered on some auction sites for items that cost up to a few hundred dollars.

nonchalant attitude in their willingness to engage in trades, when insurance is provided at low premium levels; they can undertake high risk trades with unknown parties. When such insurance is not offered, trading parties will be very careful in their trades and will not undertake that level of precarious activity. Another complication associated with providing negligible cost insurance is that it creates incentives for crooked traders to form coalitions in which they swindle the insurance agency. In spite of the potential for fraudulent trades, an insurance agency has many advantages over individual occasional traders in that due to the large number of trades that it monitors, it has the resources and experience needed to develop data bases, data collection and analysis systems, managerial and clerical procedures, and the legal muscle to detect and reduce such swindling activities to a minimal level. An economic advantage generated by low cost insurance premiums is to encourage perceived high risk honest traders with no or limited trading history to enter the on-line trading community[2]. A positive outcome of such broader participation is that in addition to reducing the impact of fraud on the level of his own activities, the trader also benefits from a higher number of traders and a higher level of dollar volumes of activity.

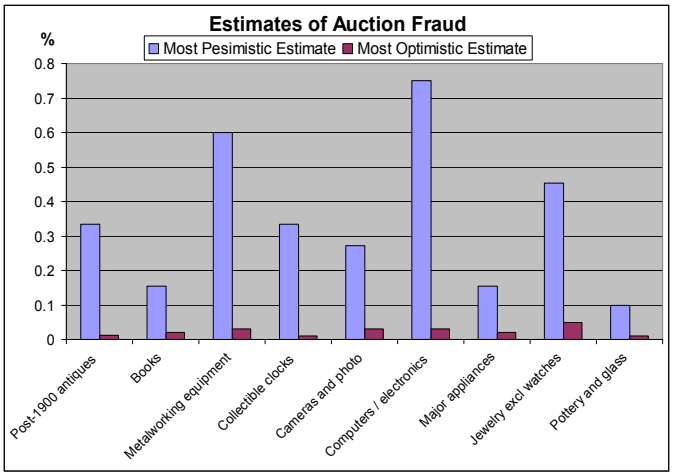


Figure 1: Estimating the number of auction based transactions that are fraudulent

In order to test the accuracy of the auction operators estimates on the level of fraudulent activity, Gavish and Tucci [5] conducted a study on the magnitude of fraudulent activities in auction sites. Obtaining accurate numbers on fraud is difficult; traders who have been defrauded are hesitant to publicize the fact that they were careless and were defrauded; Traders who were not defrauded do not have the incentive to respond to such surveys, the danger of being sued by the all powerful auction site operators is another hindrance to full disclosure of fraud information. In spite of the above difficulties and potential biases, the study established for each category of item classes a range on the number of fraudulent transactions that take place. The range estimates of the percentage of fraudulent activities are displayed in Figure 1.

The study concentrated on buyers that are being defrauded. The study discovered that sellers face a similar problem of fraud. The main difference is that auction houses have developed extensive methods for protecting sellers, but limited effort went into protecting buyers. We conjecture that it has to do with the fact that if sellers do not put items up for bid, than the auction site can't survive, while bidders/buyers on their own cannot justify the existence of an auction site. Another factor has to do with the frequency of use of the auction site. Most sellers use the auction site multiple times (in some cases, tens of thousands of times). Given a very high level of a single entity participation in auctions, justifies their investment in developing systems and methods which protect them against fraud. A buyer on the other hand has limited experience with the auction process and does not know how to protect himself against fraud, a distinct advantage for the seller.

Given the high level of fraud on the internet, it is important to develop on-line based methods for developing trust between trading partners and mechanisms to support trust developmental activities. In order to develop such methodologies we need first to understand how trust develops in the real world when individuals or corporations interact without using online faceless mechanisms. In the next section we introduce how trust is formed in non-electronic interactions and trade, followed by procedures and methods for developing trustful relations on the internet or detecting that at least one of the trading parties is trust-less trading partner, the last section provides a discussion and suggestions for further investigation.

2 How is trust created in the Real World

Before analyzing trust in an electronic environment it is important to understand the processes and mechanisms through which trust is created in different environments. They can include trust between individuals, trust between corporate entities, and trust between tribes or nations. Trust building mechanisms have to consider national versus international trust. We concentrate first on trust creation and handling between individuals. Trust is of concern, whenever at least one party has the potential to loose through actions of another party that he trusts. The loss is not limited to monetary or financial loss, it can involve decrease in reputation, embarrassment when secrets or private information is revealed, breaking secrecy in nationally secret activities, a journalist revealing his sources, or a physician or priest breaking his seal of secrecy.

[2] When tracking final bid prices on auction sites, we observed a price differential for the same item being auctioned, in favor of traders with an established history. In some cases the price differential reached up to $500 on items in the $1500 to $2000 range.

The first level of trust is created when personal direct interaction takes place between individuals, trust can be created by observing a persons behaviour towards you and others, does he show tact, is he telling stories about others, is he bragging, or does he reflect confidence and integrity. By observing his actions, having conversations with the other party and formal and informal correspondence with him trust can develop. Formal correspondence has the advantage that in some cases it can be a basis for social or legal action in case the trust is compromised.

A second layer of trust is created through repetitive transactions with an individual, mainly transactions which involve financial and commercial activities such as buying and selling, when one party is the buyer and the other is the seller. A safe policy for developing trust is to begin with low dollar volume transactions; limiting possible damages to either party. If both parties come through on the limited transactions they can increase the volume of transactions or dollar value of their activity.

Trust is especially valuable when both parties have to lose when they break their mutual trust. The loss does not have to be tangible loss, for example future losses can be generated through reputation effects, low reputation can influence others when dealing with the trader who broke financial or social trust. Such a dual effect reduces the incentives to breach trust for short term or low level gain.

Another mechanism to determine individuals and company trustworthiness is to initiate and collect information on their past trustworthiness. This can be achieved through the use of intermediaries who dealt with the other party in the past. Even when they are identified and used there are a number of possibilities:

1. If the intermediaries are personally known to the evaluator, he knows how much bias exists in their opinion, and how much weight to attach to their judgement; are they always lenient in their evaluation, or are they harsh. In addition he can collect from them objective data on the other party and their experience in dealing with him.
2. If the intermediaries and their characteristics are not known, information can be collected on their opinion worthiness, care and judgement should be used as to what weight is assigned to their opinions, and if to contact them for their input, or not to contact them at all. Such a method is used in real life when a candidate for a job supplies references. It is unlikely that he will include in his list of references, people who have a poor opinion on him. Even in such cases it is possible to overcome the initial bias by asking the references if they can recommend others who are not on his list who know him or for whom he has done work in the past. Such a method is used by security agencies, to check individuals for security clearance for sensitive positions.

The other party can provide objective data that helps in determining if you want to do business with him. For example who are or were his customers, financial data, annual reports, past delivery dates, quality measures, attrition rate of employees and managers. It is possible also to collect objective measures on past performance of the other party, such as bank records, membership in professional organizations, address and length of time in his line of business, etc.

Another avenue open to test the honesty or trustworthiness of the other party, can be achieved by proposing to him to take actions which are in the grey area of legality. For example by paying him under the table, offering a service or a favour (on the boundary of bribe). By taking the bait, the initiator collects information on the level of honesty of the other party. This is method can be a double sward method, as it can reflect badly on the entity proposing such an action. Caution has to be undertaken when offering an action whose legality is questionable.

When the trading partner is an unknown entity, it is possible to shift the burden to him, by requiring that he take share the risk or the full risk involved in the transaction. Examples include:

1. Payment after delivery, ask the other party to deliver before payment is made, i.e. ask the other party to fulfil their part of the agreement before you do your part.
2. The second party should provide a letter of credit from a bank, or an active credit card, so that he can be charged in case he is not delivering according to the agreement.

Let others share the risk (for a fee) this includes the use of escrow or insurance services. Paying a fee to a reputable agency that insures the transaction in case that the other party does not fulfil his commitment. Another mechanism in the same vein is the use of COD (Charge on Delivery) the delivery company is the one collecting the payment after the buyer has checked and verified that he received what he ordered.

Forming an opinion on the trustworthiness of someone in a commercial transaction is not easy when the two parties operate in the same city, state or country. It becomes a lot more difficult when the trading parties are in different countries and continents. Issues such as; different languages which might involve misunderstandings when translations take place, Different cultures and customs which could be misinterpreted by the other party, different legal and financial institutional arrangements, for example who has legal authority in case of disputes? How to construct a contract which is acceptable in two countries, who has the rights to intellectual property generated thru the joint

activity, Or how to take into account the impact of future exchange rates.

3 Signs that indicate con artist activity

There are a number of methods that swindlers use to defraud participants in auctions [2, 5]. We classify the main methods used:

1. *Non-delivery*: The swindler offers to sell items or provide services; he receives payment but does not deliver anything to the buyer. Here there are several sub-cases:
   a. *Delivery does not take place* at all and no refund is offered or made. The buyer is out of the amount paid.
   b. *Delivery does not take place*, but the seller offers to refund the amount paid. The buyer is happy, as the sum he paid will be returned to him. This is an interesting scheme as the swindler makes money at the expense of the buyer. There are several variants of this scheme:
      i. The sum he paid is returned to him, typically in a check that he has to deposit. Sometimes the cost of shipping/handling is deducted from the overall amount refunded. The swindler makes money at the expense of the buyer. This scheme works in the following way.
         1) The seller offers to sell an expensive item worth a few thousand dollars. He states in advance that the actual shipment of the order will take place 45 days after full payment has been received.
         2) The buyer pays for his order.
         3) The shipment does not take place within the promised time frame.
         4) After waiting some time, the buyer contacts the seller.
         5) The seller explains that due to manufacturing delays at the manufacturer site/s (or some other excuse), he could not deliver on time and offers the buyer to wait a few more weeks, or accept a full refund of his original payment.
         6) Many buyers decide to wait the extra time, the logic being, I waited two months, what is the harm of waiting two or three more weeks. Once the waiting period has passed, one of the following procedures takes place:
            a) *No shipment takes place* and the buyer is refunded his money (typically 60 to 70 days after his original payment). The buyer does not complain as his payment was refunded. In this version the seller makes money from the interest on the float, auctioning around 25 products per day, for an average final price of $2000, the amount of cash available to him for investment is $2000x25X60=$3,000,000. Using a conservative interest rate of 5 percent per year, he earns around $150,000 per year just on the interest, minus the administrative costs of receiving payments, handling the accounts and issuing refund checks, all of which can be fully automated.
            b) *Shipment takes place* of an alternate product with similar stated characteristics. Most buyers do not realize minor changes in the shipped product versus the advertised one, such as the original product was suppose to have a Mobile Pentium 2GHz. The seller delivers as a compensation for the long delay a “superior” product, with a faster processor of 2.4GHz, what the buyer does not notice is that it is a Pentium and not a Mobile Pentium; he looses mobility, and a difference of a few hundred dollars. If the buyer understands the consequences of such a switch, he can return the item for a full refund. However, based on reading the positive feedback from buyers who went through such a process, most of them do not detect or understand the difference, they accept the inferior product as a superior one and actually thank the seller for his “generous” service.
            c) *Shipment takes place* of the advertised or promised product. The sellers benefit in this case stems from the observation that in many electronic products prices decline by 3 to 5 percent per month. An item auctioned for $2000, can be bought for around $1850 two months later. The seller buys the item at the lower price and ships it to the buyer, pocketing the difference in price minus the shipping and transaction costs. In another variant of the same scheme, the seller (or his software agent) follows items auctioned or sold on multiple sites, when he discovers the same item he committed to sell, priced at a much lower price on another site; he buys it at the lower price, and ships it to the original buyer.

It is difficult to argue the illegality of the above scheme, as the buyer was a willing participant throughout the process. However, the buyer has the option of using civil lawsuit to recover the damages caused

by the delay, an option that buyers rarely exercise.

2. *Shipment and delivery takes place* but:
   a. An empty box or one containing worthless items is sent. There were cases in which stones or send was delivered, the sellers were careful to make sure that the package had the same weight as a legitimate package. Having the same weight the sellers could make the claim that they actually shipped the item. If the shipment was insured, the insurance company had to cover the cost of the item.
   b. A partial shipment or substitute items are sent, for example a laptop with a lower disk or memory capacity, or a Panel Plasma TV without a tuner or loudspeakers.
   c. Damaged or non functional merchandise are shipped, if the shipment was insured the insurance company pays for the defective items, or if the item is still under manufacturer warranty he repairs the defective item.
3. An unsolicited proposal to sell the buyer the same type of equipment he showed interest in or was bidding for, typically at a lower price than the winning bid. If the bidder responds to the solicitation he is asked to pay through Western Union, or mail a cashier's check. Once the check or transfer is cashed, nothing is mailed and the buyer is out the cash. Several variants of this method have been developed to handle suspicious buyers:
   a. If the buyer insists on using an escrow company, the seller sends back a message, agreeing to use an escrow company and provide the buyer with a link to an escrow company. Unfortunately, in several cases we checked, the escrow company was a fraudulent company set up by the swindlers.
   b. If the buyer insists on using a specific shipping company (ABC), the seller agrees to it and provides a link to a legitimate looking shipping company. Unfortunately, the only barely detectable difference is that instead of www.ABCD.com the buyer is directed to www.ABCD.org, a different suffix set up by the swindlers. In all other respects, the site appears identical to the legitimate company (the swindlers simply copy the legitimate site).
   c. Specifying that they are from the US and a legitimate business, while payment is made to an overseas account. Once payment is made, the payee can forget about receiving his order.

   The crooks in this case benefit even if no payment is made. Once the potential victim has responded to the solicitation (but decides not to transact), his name and email address are sold to spamming lists, identifying him as an active email.
4. *Planted Bidders in auctions,* Another method used in on-line auctions is for the seller to have a partner who bids up the price of an item being auctioned. This happens when the partner detects a buyer who is willing to raise the price of the item. At some price level, the partner stops bidding and the buyer pay's a price higher than what he would have paid without a planted artificial bidder. Another outcome is that the buyer stops bidding before the partner bailed out. In this case the partner is the winner of the auction. If such a case occurs one of two scenarios take place:
   a. The partner does not pay, and the item is re-listed on the same or another auction site. The seller does not complain, and no bad ratings are set on either party.
   b. The seller sends an email to the "second" highest bidder, offering to sell him the item for his highest bid price. The seller offers an excuse, such as he had several copies of the item and was willing to sell a second one, or he was cheated by the winning bidder. In many cases the bidder decides to accept the offer, as he was willing to pay that price.
5. *Fishing* is a method used by swindlers to defraud innocent users of the internet, the method is similar to hook and lure based fishing (this is the source of naming the method). When fishing the fisherman throws a hook with bait to the water, a fish (out of many potential fish) takes the bait and is hooked. Swindlers use a similar process: They send out a bait message as an email to thousands of innocent Internet users at a time.
   a. The bait message states:
      1) We are glad to inform you that you won $10,000 in a lottery, to deposit your lottery winning we need (here come several possibilities):
         a) Your banks account information so that we can perform an electronic fund transfer to your account.
         b) Your computer account information (just the password)
         c) Your credit card information…..
      2) You have an account on the following auction site/payment system, it was compromised, to make sure that yours was not compromised please enter your password on the following message coming from that site.
      3) A government agency wants to verify your information, please enter your social security number, bank accounts information,…

4) Your credit card was charged by mistake, to make sure that this was a legitimate transaction please enter your credit card number, expiration date and name

5) We can sell you at bottom prices:

a) Free cable TV, medicine, pornographic materials, offshore investments, breast enlargement procedures. Typically the swindlers sell services or items, that are illegal or embarrassing to the buyer if they become public. This decreases the likelihood that the buyer will turn to legal authorities for help.

6) The police, your ID was stolen and in order to prevent unauthorized use of your card please enter your credit card information

b. Some users bite the lure and "buy" the product or provide the information asked by the crook (the fisherman). Typically it is a very small portion (in the range of one out of a thousand) of users who accept and respond to the message.

c. The next step is for the crook to get from them their information useful to his scheme. Once the information has been collected from several unsuspected users. The swindler establishes a temporary address, orders expensive merchandise online and charges it to their credit cards.

d. The swindler asks for rush shipments (overnight if possible) of the items he purchased, once the items have been delivered he picks up the items, sells them and disappears.

e. It is difficult to track such crooks, as they use a similar process of fishing, to collect networking account and password information from users. This information is used later as the mailing address of the fishing messages; The use of the account information are typically done a long time (weeks or months) after the collection, it is used mainly to invoke the credit card scheme. Waiting for a few months reduces the chance that the account will be traceable to the crook.

A swindler using the fishing method can collect large sums of money per day even when a very small fraction of users respond and provide the information. In order to ensure a constant flow of income, the crook has to constantly send out such messages to tens of thousands of users. By doing it swindlers contribute and become a major source of spam on the Internet.

4 Establishing an appearance of legitimacy

For swindling schemes to work, swindlers have to convince a subset of users on the Internet or auction sites that they are dealing with a trustful entity. Some of the methods crooks use include: Establishing a history which generates trust, masquerading as a known trustful entity (a business, bank, government agency, escrow service).

1. An example for positive history building that swindlers use to appear as a legitimate seller to potential bidders. To establish a good track record, they put up many low cost (a few dollars each) items for sale and provide excellent service. The feedback from the buyers is highly positive. Once an excellent track record has been established, they go for the kill by putting up for bid expensive items that are never delivered. Once discovered, they repeat the same process using a different identity.
2. Some auction houses provide past rating from buyers and sellers for sellers and buyers, but in the aggregate data they do not distinguish between the seller's past selling and buying feedback. Swindlers use a policy in which they buy many items for a dollar or so. They provide excellent feedback to the sellers. This feedback appears in their aggregate statistical history, once they have established a long positive record; they put up expensive items for auction.
3. Using several IDs so there is no track record. It is easy for swindlers to receive several valid credit cards and use them to open accounts using different identities. By using the credit card information collection scheme described in the previous section, they have many cards and identities at their disposal. Having no track record, or by using the positive track record of others they are able to convince others that they are honest traders.
4. Selling for the first time (i.e. no past history exists), many users are cautious when dealing with such sellers (or buyers), however, if they offer to sell at a significantly lower price than the market price, the grid factor enters the picture and there is a good chance that a greedy trader will be hooked to this offer.
5. Some swindlers advertise items for sale and their payment terms use legitimate payment channels. However, when a customer shows interest, they are changing the payment method. For example claiming that they accept credit card payment but after the auction or trade, insisting on non-credit card payments.
6. Use of credible commercial establishments as front to their activities, there were a number of cases in which swindlers used a fake URL and web pages of a bank to swindle unsuspected users from their money, fake escrow services, fake package delivery COD services are other examples for such masquerading.

5. Recommendations

The findings in Gavish and Tucci [5] and in Snyder [7], although preliminary, indicate a source of concern for traders (sellers and buyers) on the Internet.

Even though the number of traders has been rising steadily for the last several years, it would not take much of negative media based publicity to reverse this trend [5]: traders who become fearful that they will not receive the payment, item or service they paid for, will stop participating in electronic commerce activities, which means fewer online traders, which leads to even fewer traders, and so on. To reduce the likelihood for such a vicious cycle taking place, we have several recommendations based on the above studies and observations. We believe that one of the best methods to battle swindlers is to provide information to potential traders and to educate them.

- Sites should provide as much information on traders as the law permits. Legal, privacy and security issues prevent the full disclosure of information on traders. Special attention should be paid to publicize the country, state and city in which the trader resides.
- Sites should post the percentage of positive responses relative to the number of transactions for sellers (some sites have in 2003 begun to do this) rather than displaying the absolute number of positive responses. Sites should allow users to provide more refined feedback on the transactions they just performed with a trader. Many sites limit the ranking to two or three values, restricting the users to coarse evaluations, without ability to refine their feedback.
- As mentioned above, some sellers sell (or buy) a large number of cheaper items to establish their reputation. To combat this problem, we recommend two courses of action: The first is to publish the average selling price (or even better a distribution of selling prices) of all previous items offered by the seller. Auction sites should also provide the detailed history on a trader for a much longer period of time (some sites limit the feedback period to one to three months).
- Trading sites when displaying statistical data on a trader should separate the statistical data to his buying transactions and to his selling transactions. They should also publicize the percentage of cases in which payment did not take place, and the percentage of cases in which items were re-listed.
- Trading Sites should make the use of escrow services extremely easy, and possibly mandatory. Further, the buyer/seller should only use escrow services that are certified by the auction house. To that effect, auction sites should display the list of escrow services that they have certified.
- Traders and Internet users should not respond to unsolicited offers/requests. Answering to such requests puts the user on the spam list guaranteeing that he will be bombarded with future messages.
- Better information exchange methods between auction site operators and trading sites, about swindlers and swindling schemes that they detect. They are hesitant to do it due to the legal liability when the wrong trader is identified as a swindler.
- User discussion groups are highly effective in disseminating information; as such they have the potential of curbing the swindling activity. Here again they could face the legal liability issue, and the ability of a trader to generate positive feedback by masquerading as a buyer (while he is a seller).

Another area in which users can be proactive in protecting themselves include things such as changing their passwords frequently, not providing account and credit card information to unknown entities, reporting on swindling or suspicious activities (the difficulty is that in many cases the users don't know which channel to use for reporting).

Other methods that are based on economic reasoning are developed and described in [4]. The research reported here is an ongoing research activity; By no mean is it complete, we will appreciate it if readers who discover other swindling methods or methods to prevent them will email this information to the author. We hope to incorporate what we discovered in the meantime into a future paper on the same subject.

References

[1] BBC News "The growing threat of internet fraud" November 19, (1999).

[2] J. Cassell, and T. Bickmore, "External Manifestations of Trustworthiness in the Interface", *Communications of the ACM*, vol. 43, no. 12, pp. 50-56 (2000).

[3] Cheskin, Research and Studio Archetype/Sapient Report, "E-Commerce Trust Study", http://www.sapient.com/cheskin (1999).

[4] B. Gavish, "The Economics of Spamming and Swindling on the Internet", (Electronic Commerce Research, under review).

[5] B. Gavish and C. L. Tucci, "Swindlers and Buyer dissatisfaction in Internet auctions," Working paper (September 2003).

[6] Reuters. "Scam artists make hay on online auction sites." Reuters wire service, http://www.curztech.com/news/news/2003110_06.shtml, viewed 7/13/03, (2003).

[7] Snyder, J. M. Online auction fraud: Are the auction houses doing all they should or could to stop online fraud? *Federal Communications Law Journal*, 52, No. 2, (2000) pp. 453-472.

[8] VeriSign, "Internet Security Intelligence Briefing," Report, November 10, (2003).

[9] R. Waters, "How fraudsters set traps and take the credit." *Financial Times*, IT Review, May 21, p. 1, (2003).

Learning Customer Profiles Using Unlabelled Data

Giovanni Semeraro, Pasquale Lops and Marco Degemmis
Dipartimento di Informatica – Università di Bari
Via E. Orabona, 4, I-70126 – BARI
{semeraro, lops, degemmis}@di.uniba.it

Keywords: personalization, user profiling, intelligent search and retrieval, Bayesian learning, learning from labelled and unlabelled

Received: July 25, 2003

E-commerce sites often recommend products they believe a customer is interested in buying. Many web sites have started to embody recommender systems *as a way of personalizing their content for users. This paper presents a recommender system that exploits supervised learning methods to learn user profiles from items previously rated by users. Profiles are used to find, classify, or rank items that are likely to be of interest to the user. A major concern with supervised learning techniques is that they often require a large number of labelled examples to learn accurately. Our proposal to reduce the amount of labelled data required is an algorithm that can learn effectively from a small number of labelled examples augmented with a large number of unlabelled examples. Experiments on a real dataset show that the proposed method is effective.*

1 Introduction

Across the world, 24 hours a day friends and families chat, exchange letters and pictures - all via electronic communications; business negotiates multi-million-dollar deals; products and services are bought and sold; banks process millions of financial transactions; journalists report news stories from where and as they happen; travel agents organize business and holiday trips and students research assignments. This is the Information Society. It is characterized by a very huge unstructured knowledge repository as well as a rapid increase of information. In the era of Internet, huge amounts of data are available to everybody, in every place and at any moment. This is extremely useful and exciting, but it generates anxiety, especially in novice or occasional users. From the users' point of view, obtaining the right information, which is needed to solve a problem or accomplish a task, increases the value of the Web decisively [3]. This means that the access to information is not the only relevant issue, but also the relevance of quality itself matters.

This aspect is critical in Business to Consumer (B2C) e-commerce because it is well known that the process of buying products and services often implies a high degree of complexity and uncertainty, and it is very time consuming for customers. Customers have started to require personalized support responding to their specific needs, in order to avoid the access to electronic shops that often appear like a warehouse, where you must know exactly what you want and where to find it [11]. For this purpose, many web sites have started to embody recommender systems as a way of personalizing their search process [10, 12]. Recommendation algorithms use input about a customer's interests to generate a list of recommended items. At Amazon.com, recommendation algorithms are used to personalize the online store for each customer, for example showing programming titles to a software engineer and baby toys to a new mother [2].

One of the most widely adopted approaches for performing recommendations is content-based filtering recommendation method, in which the content (e.g., text) plays an important role: the system suggests the items similar to those the user liked based on the content comparison. In a content-based system, the items of interest are defined by their associated features. This type of recommender learns a profile of the user interests based on the features present in the items the user has rated in the past [5].

A real problem with systems that learn from examples is that they require many items labelled with ratings (labelled examples). This pre-classification task is often expensive and boring for a user. In this paper, we propose, as possible solution, the use of unlabelled examples in the learning phase: the idea is to use an algorithm that learns from a few labelled examples and a large pool of unlabelled ones [4].

The paper is organized as follows: Section 2 describes a simple approach, based on the naïve Bayes machine learning method, to build user profiles for a content-based book recommender system, while the problem of learning from unlabelled data is taken into account in Section 3. An experimental session is presented in Section 4. Finally, conclusions are drawn in Section 5.

2 Learning Probabilistic User Profiles

In a previous early work [14], we presented a content-based system, called ITR (ITem Recommender), that suggests recommendations to customers of an online bookstore. ITR is a system able to recommend books by learning from their textual descriptions and ratings given by users. The system implements a naïve Bayes method to classify items as interesting or uninteresting for a particular user by exploiting a probabilistic model, learned from training examples (*preclassified instances* - the items rated by that user in the past) [6,9].

This paper presents an improved version of the system that learns from a few labelled examples and a large pool of unlabelled ones. In fact, it is very common in real applications to have situations where few labelled examples are available and unlabelled examples can be obtained without paying an additional cost. For example, in trying to learn a classifier for a user's preference for web pages, the user's bookmarks can be considered as (positive) labelled examples while unlabelled examples can be sampled from the web. In order to describe the new version of the system, we briefly recall our learning problem.

Each instance is represented by a set of slots. Each slot is a textual field corresponding to a feature of a book: *title*, *authors* and *textual annotation* (abstract). The text in each slot is a bag of words (BOW) processed taking into account their occurrences in the original text. Given a set of classes $C = \{c_1, c_2, \ldots, c_n\}$, the conditional probability of a class c_j given a document d_i is calculated according to the Bayes' theorem. In our setting, two classes exist: c_+, representing the positive class (*user-likes*), and c_-, the negative one (*user-dislikes*). Since instances are represented as a vector of three documents (BOWs), the conditional probability of a category c_j given an instance d_i is computed using the formula:

$$P(c_j \mid d_i) = \frac{P(c_j)}{P(d_i)} \prod_{m=1}^{|S|} \prod_{k=1}^{|b_{im}|} P(t_k \mid c_j, s_m)^{n_{kim}} \qquad (1)$$

where S is the set of slots, b_{im} is the BOW in the slot s_m of the instance d_i, n_{kim} is the number of occurrences of the token t_k in b_{im}. To calculate (1), the probability terms $P(c_j)$ and $P(t_k \mid c_j, s_m)$ must be estimated from the training data. Thus, each instance is weighted using a discrete rating r_i (1-10) provided by a user:

$$\omega_i^+ = \frac{r_i - 1}{9} \qquad \omega_i^- = 1 - \omega_i^+ \qquad (2)$$

The weights ω_i^+ and ω_i^- are used for estimating the two probability terms:

$$\hat{P}(c_j) = \frac{\sum_{i=1}^{|TR|} \omega_i^j}{|TR|} \qquad (3)$$

$$\hat{P}(t_k \mid c_j, s_m) = \frac{\sum_{i=1}^{|TR|} \omega_i^j n_{kim}}{\sum_{i=1}^{|TR|} \omega_i^j \mid b_{im} \mid} \qquad (4)$$

In (4), the denominator denotes the total weighted length of the slot s_m in the class c_j.
In the next section, we describe how we have used this model for learning from labelled and unlabelled examples.

3 Including Unlabelled Data Using EM

The process of learning from textual descriptions has the drawback of requiring many examples labelled with ratings. This preclassification task is often expensive and boring for a user. As a consequence, the approach is unfeasible in real world applications due to the difficulty to obtain a considerable number of items rated by users in order to learn an accurate classifier. A possible solution comes from the use of unlabelled examples in the learning phase. The idea is to use an algorithm that learns from few labelled examples and a large pool of unlabelled ones, combining the *Expectation-Maximization (EM)* technique [1] with the naïve Bayes classifier, as in [7]. The schema of the implemented algorithm is given in Fig. 1. Our prototype, called EMIR (**E**xpectation **M**aximization in **ITR**), integrates ITR with the EM algorithm, and has been used to evaluate the approach in the area of recommender systems, where the problems of obtaining user ratings is of primary importance.

Input:

D_i^l = items rated in a specific category by user U_i

D_i^u = items not rated by the user U_i in the category

1. Train ITR using only labelled documents D_i^l
   Parameters are estimated as in Eq. (3) and (4). Obtain a classifier NB_0
2. Loop while classifier parameters change
   (E-step): Use the current classifier NB_k to compute $P(c_+ \mid d_j)$ for each d_j in D_i^u (Eq. (1))
   (M-step): Re-estimate classifier parameters using Eq. (3) e (4) given the probabilistically assigned class for d_j

Figure 1: Algorithm for learning user profiles using EM.

4 Experimental session

In this section the experiment we performed in order to evaluate the effectiveness of the proposed method is described.

4.1 Experiment Description

For the experiment, 5 book categories were selected at the Web site of a virtual bookshop. For each book category, a set of book descriptions was obtained and stored in a local database. Table 1 gives the specific figures for each category in terms of number of books available, number of books with a textual annotation, and average length (number of words) of the textual annotation. Each user involved in the experiment was requested to choose one category of interest and to rate 100 books, providing discrete ratings between 1 and 10. In this way, a dataset for each category was obtained. From each dataset D_i, 70 examples were randomly selected and used as training set TR_i; the rest of D_i was used as test set TS_i.

Table 1: Database Information.

Dataset	# of Books	# of Books with abstract	Abstract length (avg. value)
Computing & Internet	5414	4190 (77%)	42.39
Fiction & literature	6099	3378 (55%)	35.54
Travel	3179	1541 (48%)	28.29
Business	5527	3668 (66%)	42.04
SF, horror & fantasy	667	484 (72%)	22.33
Total	20886	13261 (63%)	

4.2 Experiment Description and Evaluation Measures

An experiment was performed to compare the performance of EMIR with respect to ITR. Specifically, the main objective of the experiment was to verify whether EMIR could reach the same performance of ITR using a fewer number of labelled instances and exploiting a large pool of unlabelled examples (all books not rated by the user).

For each training set TR_i, we randomly selected n labelled instances to build two distinct classifiers, one running ITR and the other running EMIR. The former was induced just from the n labelled documents, while all the available unlabelled instances were also used for the latter. The training phase was repeated 80 times by varying n in the set {5, 10, 20, 30, 40, 50, 60, 70} and, for each one of the 8 values of n, by repeating 10 times the random choice of the n labelled instances for the TR_i. Several metrics were used in the testing phase.

Since the task was to identify or retrieve items preferred by users from a repository, traditional information retrieval measures were adopted, namely *Precision (Pr)*, *Recall (Re)* and *F-measure (F)*, a combination od Precision and Recall [13]. Moreover, we also adopted the *Normalized Distance-based Performance Measure (NDPM)* [15] to evaluate the goodness of the items' ranking, calculated according to a certain relevance measure. Specifically, *NDPM* was exploited to measure the distance between the ranking imposed by the user ratings and the ranking predicted by the system. Values range from 0 *(agreement)* to 1 *(disagreement)*.

4.3 Analysis of results

For the sake of simplicity, we present only results for the datasets "Computing and Internet" and "Travel". Figures 2 and 3 depict the effect of varying the amount of labelled data. The number of unlabelled instances is constant, while the number of labelled instances in the horizontal axis varies. The values of measures are calculated by averaging the results of the 10 trials as described in Section 4.2. The results show that the use of EMIR improves Recall, F-Measure, and NDPM. For example, the highest Recall value reached by ITR is 69% (using 60 labelled instances), while EMIR reaches 96% using only 20 labelled instances. Again, EMIR dominates ITR as regards NDPM. As regards Precision, it seems that adding unlabelled data to a small amount of labelled data hurts performance. These results will be analyzed further.

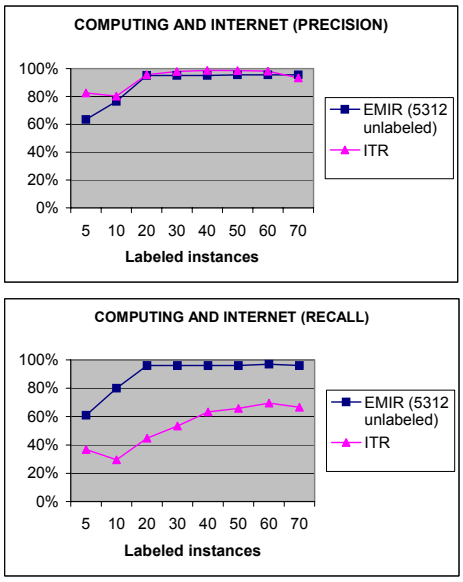


Figure 2: Precision and Recall of ITR and EMIR on the "Computing and Internet" dataset.

We now move on to the "Travel" dataset. Results in Figures 4 and 5 show that EMIR reaches the highest value of Precision using 20 labelled data, while ITR needs 50 labelled instances to obtain the same result. EMIR exceeds 80% Recall using 30 labelled instances, while ITR never achieves this result.

To sum up, EMIR outperforms ITR, as shown by values of F-Measure. Finally, values of NDPM seem similar when the number of labelled instances is less than 30, but surprisingly, ITR outperforms EMIR when the

number of labelled instances is greater than 30. We think that this is probably due to the distribution of ratings given by the user. In fact, the average rating for the "Computing and Internet" dataset is 6.55 and 70% of training documents is positive (rating from 6 to 10), while for the "Travel" dataset the average rating is 6.33 and 66% of training documents is positive. Probably, this is not the only reason for such a behavior; we will further investigate on that.

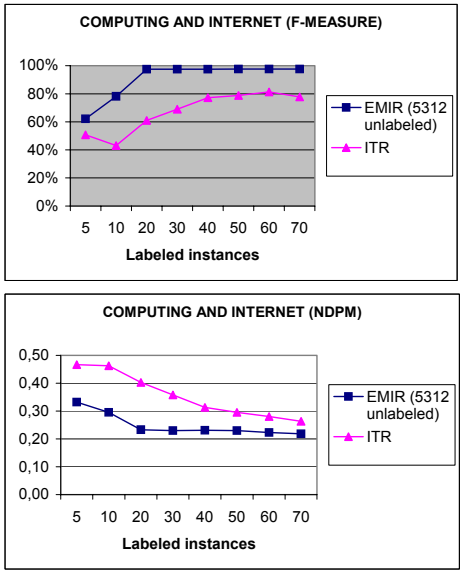


Figure 3: F-measure and NDPM of ITR and EMIR on the "Computing and Internet" dataset.

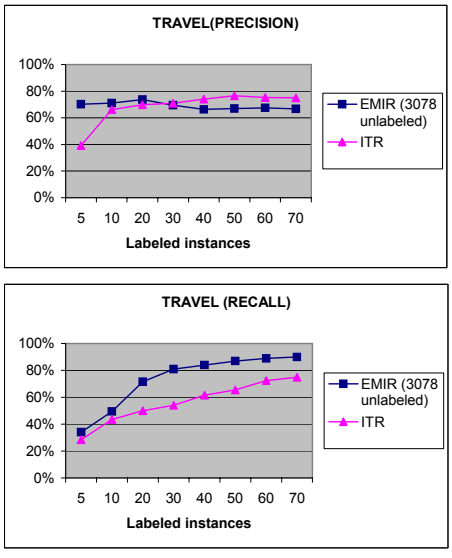


Figure 4: Precision and Recall of ITR and EMIR on the "Travel" dataset.

In order to verify if the obtained results are statistical significant, we adopted the non-parametric Wilcoxon signed rank test [8], since the number of independent trials (i.e., datasets) is relatively low and does not justify the application of a parametric test. The test was adopted in order to evaluate the difference in effectiveness of the two systems according to the performance metrics described in Section 4.2. The test compared values obtained by ITR when trained using 70 labelled instances, with values obtained by EMIR trained with a set of n labelled instances, n varying from 5 to 70.

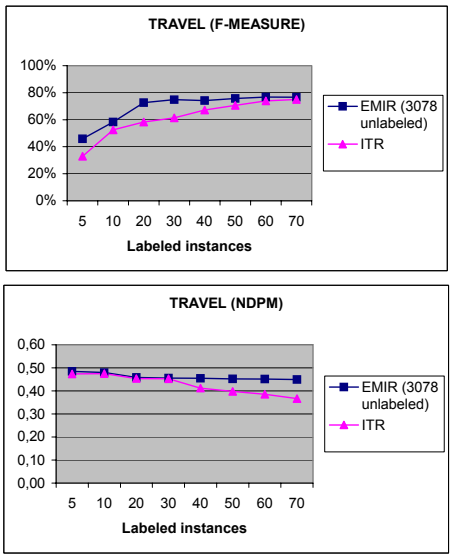


Figure 5: F-measure and NDPM of ITR and EMIR on the "Travel" dataset.

Differences in performance are statistically significant ($p < 0.05$) only for *Precision*, in favour of ITR. This means that it is possible to obtain reliable recommendations using only 5 labelled instances, even if this leads to a loss in precision. The same results are obtained setting $n = 10$. Tables 3, 4 and 5 report the summary of results with $n = 20$, 30, 40 respectively. Notice that with $n = 20$, there is no difference ($p < 0.05$) between the systems. With $n \geq 30$, the results show that EMIR performs at least as well as ITR. Indeed, we observe that Recall calculated by EMIR is significantly higher than the one calculated by ITR and this difference is statistically significant. These results indicate that the number of labelled instances to obtain a reliable classifier for item recommending can be dramatically reduced using a large amount of unlabelled data.

Table 2: Results of the comparison between ITR (70 labelled) and EMIR (5 labelled)

	Precision		Recall		F-Measure		NDPM	
Dataset	ITR	EMIR	ITR	EMIR	ITR	EMIR	ITR	EMIR
Computing & Internet	0,933	0,635	0,667	0,610	0,778	0,622	0,263	0,332
Business	0,667	0,418	0,706	0,641	0,686	0,506	0,438	0,357
Travel	0,750	0,703	0,750	0,340	0,750	0,458	0,367	0,485
SF, fantasy & horror	0,875	0,722	0,333	0,348	0,483	0,469	0,436	0,545
Fiction & literature	0,400	0,295	0,100	0,285	0,160	0,290	0,640	0,398
Avg.	0,725	0,554	0,511	0,445	0,571	0,469	0,429	0,423
W=	**15**		**5**		**11**		**-1**	

Table 3: Results of the comparison between ITR (70 labelled) and EMIR (20 labelled)

	Precision		Recall		F-Measure		NDPM	
Dataset	ITR	EMIR	ITR	EMIR	ITR	EMIR	ITR	EMIR
Computing & Internet	0,933	0,950	0,667	0,960	0,778	0,975	0,263	0,233
Business	0,667	0,662	0,706	0,818	0,686	0,732	0,438	0,383
Travel	0,750	0,737	0,750	0,715	0,750	0,726	0,367	0,458
SF, fantasy & horror	0,875	0,746	0,333	0,500	0,483	0,599	0,436	0,463
Fiction & literature	0,400	0,422	0,100	0,200	0,160	0,271	0,640	0,420
Avg.	0,725	0,703	0,511	0,639	0,571	0,660	0,429	0,391
W=	**1**		**-13**		**-13**		**5**	

Table 4. Results of the comparison between ITR (70 labelled) and EMIR (30 labelled)

	Precision		Recall		F-Measure		NDPM	
Dataset	ITR	EMIR	ITR	EMIR	ITR	EMIR	ITR	EMIR
Computing & Internet	0,933	0,950	0,667	0,960	0,778	0,975	0,263	0,230
Business	0,667	0,659	0,706	0,847	0,686	0,741	0,438	0,378
Travel	0,750	0,695	0,750	0,810	0,750	0,748	0,367	0,456
SF, fantasy & horror	0,875	0,776	0,333	0,538	0,483	0,636	0,436	0,459
Fiction & literature	0,400	0,741	0,100	0,160	0,160	0,263	0,640	0,428
Avg.	0,725	0,764	0,511	0,663	0,571	0,672	0,429	0,390
W=	**1**		**-15**		**-13**		**5**	

Table 5. Results of the comparison between ITR (70 labelled) and EMIR (40 labelled)

	Precision		Recall		F-Measure		NDPM	
Dataset	ITR	EMIR	ITR	EMIR	ITR	EMIR	ITR	EMIR
Computing & Internet	0,933	0,950	0,667	0,960	0,778	0,975	0,263	0,231
Business	0,667	0,659	0,706	0,900	0,686	0,761	0,438	0,363
Travel	0,750	0,663	0,750	0,840	0,750	0,741	0,367	0,454
SF, fantasy & horror	0,875	0,746	0,333	0,514	0,483	0,609	0,436	0,450
Fiction & literature	0,400	0,959	0,100	0,225	0,160	0,364	0,640	0,441
Avg.	0,725	0,796	0,511	0,688	0,571	0,690	0,429	0,388
W=	**1**		**-15**		**-13**		**5**	

5 Conclusions

In this paper an approach to infer user profiles for item recommending was discussed. We presented a process that turns unstructured data (textual product descriptions) into knowledge about customers' preferences using supervised machine learning techniques. In particular, a naïve Bayes classifier was used to construct a model of user's interests based on user ratings. However, obtaining training labels is often expensive, thus we proposed an approach based on the combination of EM and naïve Bayes classifier, that can exploit unlabelled documents in the training process. Experiments provide evidence that the quality of recommendations is improved.

Acknowledgment

The authors would like to thank Daniele Capursi and Roberto Scannicchio for their enthusiasm and dedication in designing and developing ITR and EMIR.

References

[1] M. M. Dempster, N. M. Laird, D. B Jain (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of Royal Statistical Society,* Series B, 39, pp. 1--38.

[2] G. Linden, B. Smith, J. York (2003). Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, 7(1), pp. 76--80.

[3] P. Maes (1994). Agents can reduce work and information overload. *Communication ACM*, 37(7), pp. 30--40.

[4] T. Mitchell (1997). *Machine Learning*. McGraw-Hill, New York.

[5] D. Mladenic (1999). Text-learning and related intelligent agents: a survey. *IEEE Intelligent Systems*, 14(4), pp. 44--54.

[6] R. J. Mooney and L. Roy (2000). Content-based book recommending using learning for text categorization. *Proceedings of the 5th ACM Conference on Digital Libraries*, pp. 195--204, San Antonio, US. ACM Press, New York, US.

[7] K. Nigam, A. K. McCallum, S. Thrun, T. M. Mitchell (2000). Text Classification from Labelled and Unlabelled Documents using EM. *Machine Learning*, 39(2/3), pp. 103--134.

[8] M. Orkin, R. Drogin (1990). *Vital Statistics*. McGraw-Hill, New York .

[9] M. Pazzani, D. Billsus (1997). Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine Learning*, 27(3), pp. 313--331.

[10] P. Resnick, H. Varian (1997). Recommender Systems. *Communication ACM*, 40(3), pp. 56--58.

[11] B. M. Sarwar, G. Karypis, J. A. Nonstan, and J. T. Riedl (2000). Analysis of recommender algorithms for e-commerce. *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pp. 158--167.

[12] J. Schafer, J. Konstan, J. Riedl (2001). E-Commerce Recommendation Applications. *Data Mining and Knowledge Discovery*, 5(1/2), pp. 115--153.

[13] F. Sebastiani (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1), pp. 1--47.

[14] G. Semeraro, M. Degemmis, P. Lops (2002). User Profiling to Support Internet Customers: What Do You Want to Buy Today? *Informatica Journal,* 26(4), pp. 407--418.

[15] Y. Y. Yao (1995). Measuring Retrieval Effectiveness Based on User Preference of Documents. *Journal of the American Society for Information Science*, 46(2), pp. 133--145.

Synergetic Integration of Aglets and E-speak in E-Commerce

Andreas Schmid and Tong-Seng Quah
Information Communication Institute Singapore, School of EEE, Nanyang Technological University
Nanyang Avenue, Singapore 639798, Republic of Singapore
E-mail: Andreas.Schmid@ieee.org

Keywords: E-speak, Aglets, E-Commerce, Mobile Agent, Brokering, E-Services, Integration, Synergy

Received: July 25, 2003

Our project proposes and implements a system for seamless integration of HP E-speak E-services and IBM Aglet mobile agent applications onto a common platform. The developed software enables all applications written in Aglets and E-speak to create joint services across the two platforms. It establishes an environment for collaboration, intercommunication, translation, and bridging. The whole design is thereby highly reusable and portable. This paper also analyses the requirements of E-Commerce and elaborates how effectively a combination of Aglets and E-speak can provide the necessary resources. We suggest a common model for Aglets, E-speak, and traditional Internet components that enhances performance through synergies. The integration combines the advantages of each technology in its respective domains.

1 Introduction

An Aglet is a Java-based autonomous mobile software agent. It is a program that can halt itself, ship itself to another computer on the network, and continue execution at the new computer [9]. The agent doesn't restart execution from the beginning at the new computer; it continues where it halted on the last host. The key feature is that both its code and state are mobile. Aglets are autonomous because they control their own lifecycle, receive requests from external sources, and decide to perform actions independently [8].

E-speak is an open-source software platform for creating, composing, mediating, managing, and accessing Internet-based E-services, developed by Hewlett Packard [2]. These E-services can interact with each other to advertise capabilities, discover other E-services, and ally with each other to offer new functionality. E-services can even negotiate to broker, bill, manage, and monitor each other. This is all accomplished in a dynamic, platform-independent, ad hoc, yet secure manner. E-speak-enabled services can communicate through public networks, past network firewalls, and to non-networked devices through low-level protocols [1].

The process of defining how to implement multi-enterprise E-business models becomes increasingly complex and difficult to manage. Many activities to solve these problems have been undertaken with the help of mobile software agents. Until now, however, most of these projects have been stand-alone implementations, unable to interact with each other.

2 Benefits of the Integration of Aglets and E-speak

2.1 Characteristics of E-speak

At the core of E-speak is a service brokering architecture, which mediates between service providers and the customers [1]. It provides a dynamic ecosystem of E-Commerce entities offering Internet-based services. A typical request would be broken down into simpler requests, for each of which the set of service providers is dynamically discovered [4]. Bids are requested from each of them and the one best matching is selected. A final service package is created, whose delivery and billing is negotiated. This E-broker discovers, composes, and completes a serious of transactions ‘on-the-fly’. Contributed services include event aggregation, authentication and encryption, metering and billing, and firewall traversal [5].

E-services could be a travel agency, an airline, a hotel in Manhattan, a limousine service, and a flower delivery. A customer may want to visit Manhattan with his wife, having a limousine picking them up at the airport and driving to the hotel, where he wants a room higher than the eighths floor and view to Miss Liberty, while flowers for his wife are delivered to his room right after arrival.

This highly complex request could be entered to the travel agency E-service. This could lead to a dynamic discovery of the airline, limousine, hotel, and flower E-service individually and ‘on-the-fly’ combine them into the complex service request. Then, the travel agent service would bill the businessman for the whole bundle and cover the costs for its other requested services. E-speak additionally provides the functionality of

mediation. The travel agency E-service could monitor the airline service and discover that the flight was actually delayed. It would automatically reschedule all the other services and inform the customer through his hand phone about the rearrangements.

Additionally, E-speak is compatible with major E-Commerce frameworks that include CommerceNet, RosettaNet and Microsoft BizTalk Server as well as with major E-Commerce standards that include OAG (Open Applications Group), OMG (Object Management Group), OFX (Open Financial Exchange), W3C (World Wide Web Consortium) and TOG (The Open Group) [2].

2.2 Information Exchange in E-Commerce, Aglets and E-speak

Many of today's E-Commerce applications include complex business processes with a large number of concurrent tasks. These tasks may persist for a long duration, may require long waiting times, and could be nested within other tasks. Additionally they are highly asynchronous, expose continues changes and may configure on the fly.

Thus, any flat conversation management, like message exchange, lacks the scalability for handling and tracking such sizable applications. Unfortunately, message exchange is the way Aglets interact [8]. Any more complex transactions in Aglets are usually implemented through a centralized scheduling architecture, where one Aglet host serves as a coordination unit. This may work well within one single enterprise, but causes serious problems for inter-enterprise transactions.

E-speak, on the other hand, evolved from the Distributed Computing paradigm. A typical complex E-speak request is broken down into simpler requests. The set of service providers for each of these simple requests is then dynamically discovered. Subsequently, the best match is invoked, and its execution mediated [4].

2.3 Bulk Data Transfer in E-Commerce, Aglets and E-speak

As personalized, continuously running and semi-autonomous entities, Aglets communicate via message exchange [8], which may not be suitable for bulk data exchange. Routing and caching a large amount of data imposes a considerable burden for Aglets. For example, moving data between an operational database and a data warehouse via an Aglet is very unlikely. It seems necessary to separate bulk data transfer and Aglet cooperation into different environments.

E-speak can provide asynchronous and synchronous communication in the same environment [3]. Bulk data transfer is an easy task for E-speak as well as for other distributed computing environments, like CORBA and RMI. It fits closely into distributed computing and is a direct extension from Networking Transport Protocols (like TCP/IP).

2.4 XML as Joint Communication Language in Aglets and E-speak

In today's technical world many different domain specific ontologies are used. Ontology refers to the common vocabulary and agreed semantics specific for a subject domain. Currently, XML is in the process of solving this problem. Through the use of DTD (Document Type Definition) each sector can create its own semantic that fits individual needs and yet remains generally usable across sector boundaries. Consequently, E-speak provides support for XML in its API [6].

Aglets communication through messages could adopt the XML format for its message payload and conform with communication in modern E-Commerce and most new Internet applications.

The software developed during this project implements a proxy server between the Aglet world, E-speak and the Internet [10]. A DTD-based interpreter would enable document-driven Aglet cooperation. Moreover, it would allow Aglets to share ontologies for multiple or even dynamic domains. In this way, the cooperation of dynamic Aglets would support *plug-and-play commerce*; mediating businesses that are built on one another's service. Aglets would acquire some of the key functionalities of E-speak.

2.5 Firewalls in Aglets and E-speak

Internet-based E-Commerce involves multiple enterprises separated by firewalls, having self-interests and individual data sharing scopes. They need support for peer-to-peer interactions instead of conventional centralized workflow systems for E-Commerce automation.

One difficulty for the Aglet technology to fit into this picture consists in the limitation of its coordination model, which assumes a *coordinator* in each group for facilitating naming and resource services [7]. It is unlikely for the Aglets belonging to different enterprises, to form a single Aglet group, or domain to facilitate coordination across enterprise boundaries.

E-speak on the other hand has Firewall Traversal and fine-grained access control as one of its Standard Services. The E-speak Engine can be inserted at multiple points in the chain between clients and remote services [5]. These remote services act and look just like a local service, since the E-speak Engine acts like a kernel [2]. Thus, the administrator can see and control access to services inside his network and firewall traversal is supported.

2.6 Collaboration in E-Commerce, Aglets and E-speak

An E-Commerce scenario typically involves the following activities: identifying requirements, brokering products, brokering vendors, negotiating deals, or making purchase and payment transactions [8]. Mobile agents, driven by a set of beliefs, desires and intentions

(BDI) could be used to *mediate* users and servers to automate a number of the most time-consuming tasks in E-Commerce, with enhanced parallelism.

However, the previous "proof-of-concept" efforts do not scale well in E-Commerce automation [9]. An essential reason is that the agent infrastructure is primarily designed for intra-enterprise, group-based mobile agent cooperation. To my knowledge, there is no sizable, inter-enterprise E-Commerce application using a mobile agent system ever deployed.

The E-speak Service Framework Specification (SFS) defines standard business interactions and conventions as XML documents that allow E-services to dynamically discover and negotiate with each other and compose themselves into more complex services [4]. Unlike traditional software component frameworks, E-speak's SFS defines loosely coupled asynchronous messaging based on the XML document exchange model.

2.7 Dynamics in E-Commerce, Aglets and E-speak

E-Commerce applications involve a large number of heterogeneous information sources with evolving contents. Dynamic relationships among a large number of autonomous service requesters, brokers and providers are common. To support such dynamics, an E-Commerce infrastructure must support the cooperation of loosely coupled E-Business systems [9]. Aglets with predefined functions but without the ability to modify their behavior dynamically may be too limited for mediating E-Commerce applications properly. They cannot adjust their behavior to participate in dynamically formed partnerships.

Turning Aglet cooperation from conversation-level to process-level could be a solution. In general, businesses collaborate following certain business processes. Such business collaboration usually involves multiple agents, each responsible for managing or performing certain tasks that contribute to the process. Adding inter-enterprise cooperative process management capability into agent-based systems is critical for these business collaborations.

E-speak's Service Framework Specification (SFS) creates an open service model, allowing all kinds of digital functionality to be delivered through a common set of APIs [4]. SFS presents a uniform service abstraction and mediated access. New service types and semantics can be dynamically modeled using the common service representation of an E-speak resource.

2.8 A Combined Model for Aglets and E-speak

Many E-Commerce applications include complex business processes and involve multiple enterprises. To handle such applications through simple Aglet messaging is not scalable and through centralized servers is unreasonable. One approach is to lift Aglet cooperation from the messaging level to the process level, and from centralized process management to peer-to-peer cooperative process management.

These goals could be achieved through the use of E-speak and XML in combination with Aglets. The use of E-speak could allow Aglets to communicate across enterprise boundaries, with fine-grained access control, firewall traversal and other infrastructure services. E-speak could provide a messaging service to each Aglet domain coordinator within one enterprise. This service would then become the single gateway to the Aglet domain, and could be made standard for all Aglet domains. Within a domain, the domain coordinator could forward messages to other Aglets using intra-domain Aglet messaging.

This environment and architectural framework was developed during this project and the source code is available upon request [10]. This project implemented, among other functionalities, the communication in the publish/subscribe mode between any number of E-speak cores and Aglets. By using this feature, when an Aglet intends to buy some electronic parts, e.g. instead of checking the vendor Aglets one by one, it can publish an availability-check message, and E-speak can broadcast this message to the entire mass of vendor Aglets who subscribe to this message. This works easily across enterprise boundaries and across firewalls.

Modern telephone infrastructures perform a separation of the signal network and the voice trunks. The same model could be applied to an environment where Aglets and E-speak are present. Aglets' messaging network together with the above mentioned E-speak backbone for cross-enterprise messaging could then build the cooperation infrastructure. To transfer large amounts of data, for example, moving data between an operational database and a data warehouse, direct connections through an E-speak service bus could also be established in this scenario. This would ensure the required data throughput for E-Commerce. A separation in control information and data of business processes may lead to more efficiency and eventually also to more privacy.

Finally, XML might be used as *lingua franca* to enable compatibility between Aglets, E-speak and other Internet-based services. Aglets could deploy the XML format as a standardized way to represent their message payload, while E-speak provides build-in support in form of an XML API, implemented through E-speak's WebAccess. This might lead to compatibility even with many other technologies, as XML is expected to become a universal standard.

3 Architectural Overview for an Integration of Aglets and E-speak

During this project, a software collection of utility methods necessary to perform collaboration between Aglets and E-speak was successfully developed. The provided software can be used for every existing application written in E-speak or Aglets to create common services together with any number of other

Aglet and E-speak applications. It is designed with a strong emphasis for reuse and portability and is, therefore, implemented platform-independently in a highly modular way [10].

The system consists of three sub-systems: E-speak client, Aglet client and Bridge Manager. The E-speak client operates in a pure E-speak environment and provides access for external E-speak legacy applications. The Aglet client is hosted in pure Aglet environment (i.e. on a Tahiti server) and serves as a gateway for other Aglets. The Bridge Manager is the heart of the system. As combined unit of the E-speak's and Aglets' environments, it communicates and interacts with both technologies and constantly translates and bridges between the two systems. This entity allows collaboration, intercommunication, translation, and bridging between every E-speak and Aglet application. The three subsystems are shown in Figure 1.

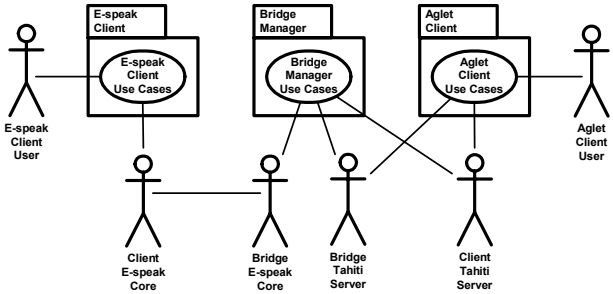


Figure 1: Overview of the Use Cases

Figures 2 to 4 provide a detailed view of the use cases contained in each individual subsystem.

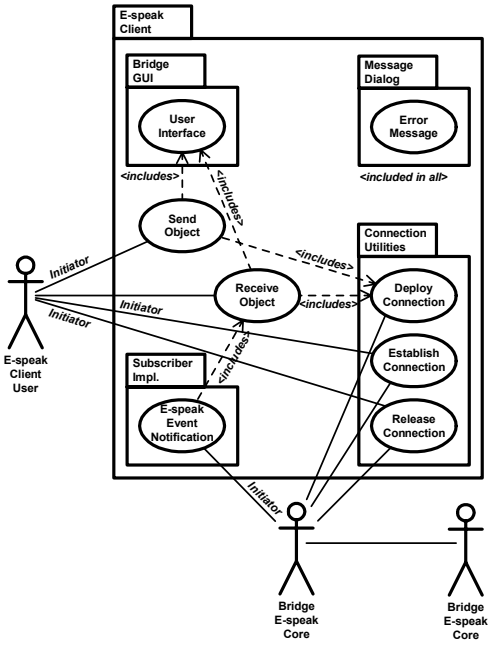


Figure 2: E-speak Client Use Case Diagram

E-speak Client software:
The E-speak Client environment includes a Client E-speak Core where the E-speak Client software connects to and runs on top of it. The main purpose is to provide access for external E-speak legacy applications to the collaboration functionalities provided by the Bridge Manager system.

Aglet Client software:
The Aglet Client software operates from inside a Client Tahiti Server and exports software interfaces to Aglet legacy applications. This system mainly serves as a gateway for Aglets to deploy the Bridge Manager's collaboration services.

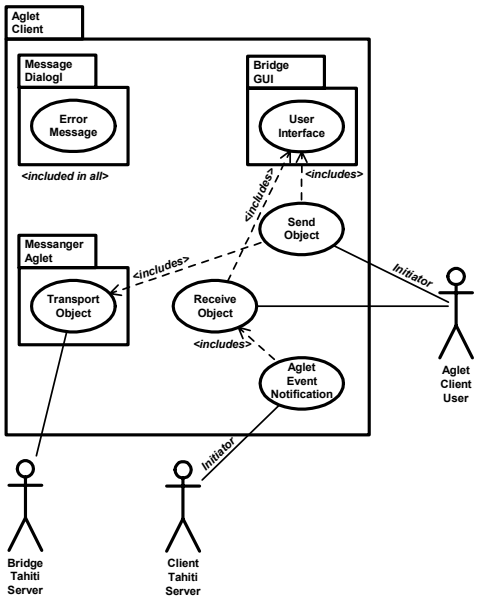


Figure 3: Aglet Client Use Case Diagram

Bridge Manager Software:
The Bridge Manager consists of both, a Bridge E-speak Core and a Bridge Tahiti Server. This combined unit together allows collaboration between any E-speak and any Aglet environment. The operating system hosting this entity needs to have both, the configuration required for the E-speak environment and the configuration required for the Tahiti Server and Aglets environment.

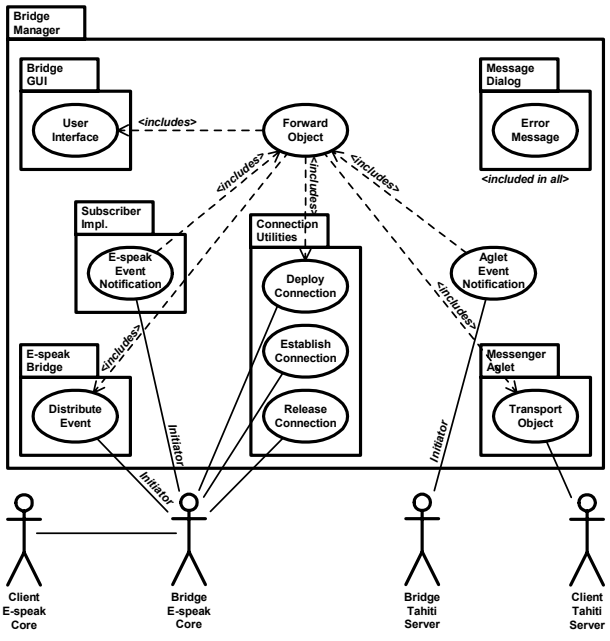


Figure 4: Bridge Manager Use Case Diagram

There can be many instances of E-speak Client systems, Aglet Client systems and Bridge Manager systems. Collaboration between all those entities will still be maintained and they can form different communities of collaboration. Multiple memberships in different communities, as well as dynamic entries and leaves are also possible [10].

4 Design Principles for an Integration of Aglets and E-speak

4.1 Choosing the E-speak Interface

Two interface options are available with E-speak:

J-ESI:
This interface to E-speak is based on Java and allows interaction with the E-speak core or E-speak services through APIs (Application Programming Interfaces) [3]. Computational Services fit well with this API style model. The API model typically assumes knowledge of the exact interface at programming time; usually through importing the IDL (Interface Definition Language) definitions at compile time to generate the stubs needed.

WebAccess:
This interface to E-speak is based on XML (Extensible Mark-up Language) [6]. Informational, business or broker type services fit well with the document mode. The client can discover changes or extensions in the document model when he downloads the schema (DTD; Document Type Definition).

4.2 Overview of Creating E-Services

The procedure of building and deploying an E-service with J-ESI involves three main steps:

- Specification of the interface for the service.
- Writing the implementation code for the interface.
- Deploying the service.

The interfaces used with J-ESI (Java E-speak Service Interface (Java-API)) have to conform to the E-speak IDL (Interface Definition Language) [3]. The next step is to specify relevant attributes and use the vocabulary service to describe the E-service. The E-services can be discovered across multiple groups of E-speak Cores. Interactions between E-services are mediated by the E-speak infrastructure [2].

4.3 The Service Contract

The service contract (interface) is defined as an E-speak IDL, which is similar to the Java-RMI IDL For every method defined in the interface, the stub class contains the code to create messages, marshal parameters, and send it to the service provider [3].

4.4 The Event Model

E-speak's Event Service is an extensible service targeted at loosely coupled, distributed applications. Events provide a publish-subscribe mechanism for communication built on top of E-speak messaging [4]. A Publisher is an entity that generates an Event notification message. The recipient of an Event notification is called a Listener. A Distributor is an extension of a Listener. It receives Events and forwards them to other Listeners. A Subscriber is an entity that registers interest in a particular Event with a Distributor and designates the Listener to whom Events are sent.

4.5 Communication in Aglets

Inter-Aglet messaging requires an Aglet to implement handlers for the kinds of messages that it is supposed to understand [7]. The message-handling method is not directly called, but a proxy serves as a message gateway for the Aglet and therefore provides a location-independent interface for sending messages to Aglets. Messages in Aglets can be any type of objects.

4.6 Aglet Collaboration through Proxies

An Aglet is fundamentally a mobile event and a message handler. Associated with each Aglet is a proxy object that acts (1) as a shield to avoid uncontrolled access to the Aglet's public methods and (2) as a convenient handle for a local, remote, or deactivated Aglet [8].

5 Summary of Use Cases for Integration of Aglets and E-speak

5.1 'Distribute Events'

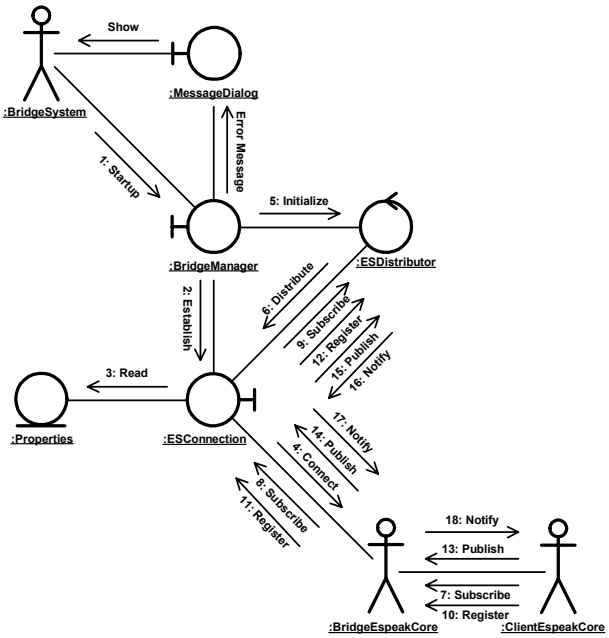


Figure 5: Distribute Events Collaboration Diagram

When the Bridge Manager software is started, it automatically initializes the Bridge E-speak Core. Then, it establishes a connection to the Core and creates an instance of the E-speak Event Distributor feature. After the Event Distributor is registered with the given

connection all results are verified. Any E-speak Client being properly connected to an E-speak Client Core can register as a publisher and subscribe for notification for a given event type. This is done by the E-speak Client through invoking the Manage Connection use case. Whenever a registered publisher sends an E-speak event notification, the E-speak Event Distributor feature will forward this event to all registered E-speak subscribers. These events can carry any type of object as payload. This feature implements forwarding of objects among E-speak Client Users.

5.2 'Establish Connection'

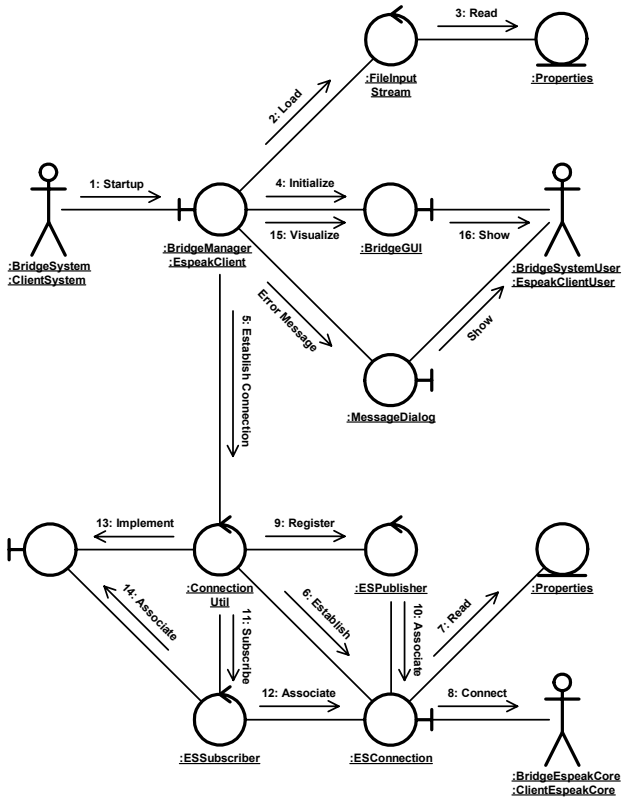


Figure 6: Establish Connection Collaboration Diagram

Whenever the start-up routine of our system is invoked, it triggers the Client E-speak core or the Bridge E-speak Core respectively to initialize. The Client E-speak core or the Bridge Manager Core and the E-speak Client software or the Bridge Manager software respectively then establish an E-speak connection to the Client E-speak Core and from there to the Bridge E-speak Core or respectively directly to the Bridge E-speak Core. The connection parameters are read in from the corresponding property file. The E-speak Client software or the Bridge Manager software then register with the Bridge E-speak Core as the E-speak event publisher for the given event type. The E-speak Client software or the Bridge Manager software subscribe as receiver for E-speak event notifications of the given event type. All results are continuously verified and if necessary the corresponding exception handling is applied.

5.3 'Send E-speak Object'

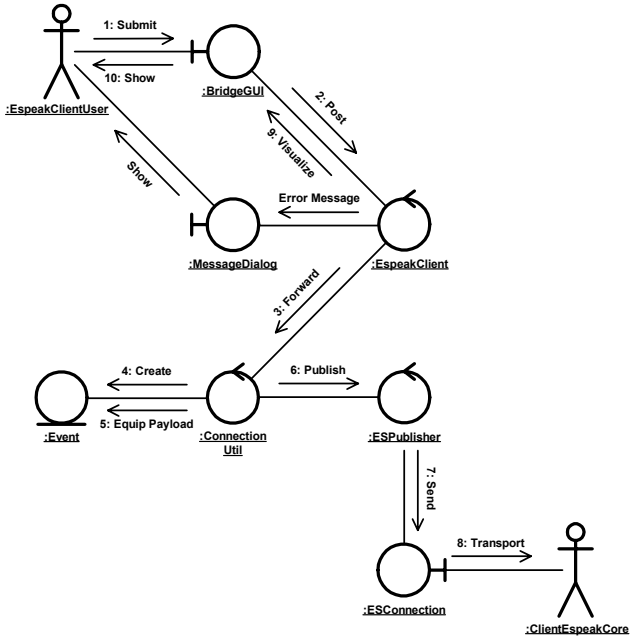


Figure 7: Send E-speak Object Collaboration Diagram

After the E-speak Client software has successfully initialized; the Client GUI is presented, and the Client E-speak Core is set up and running, an E-speak object can be send. The Client E-speak Core has therefore already established a connection, registered as publisher, and subscribed for notifications with the Bridge E-speak Core by invoking the Establish Connection use case. The Client software can now send any kind of object to the Bridge Core. This object is attached as payload together with the notification for an E-speak event. At the Bridge E-speak Core, the Distribute Events use case will take care of the distribution to all E-speak Client Users. Additionally, the Notify E-speak Event will take care of the distribution to all Aglet Client Users. As usual, all results are continuously verified and respective exception handling applied. The object supplied from the E-speak Client User will thus be forwarded to all other E-speak Client Users and all other Aglet Client Users registered for notification.

5.4 'E-speak Event Notification'

This use case can be invoked after the E-speak Client software has successfully initialized and the Client E-speak Core has established a connection, registered as publisher, and subscribed for notifications. The Bridge E-speak Core then invokes this use case when it connects to the Client E-speak Core and sends an E-speak event notification to the E-speak Client software. The object provided together with the E-speak event notification message will be extracted and its object-type identified. The Receive Object use case will be invoked to handle the respective object-type. That use case will display the received result in the according way inside the Client GUI and make the object available for external software. The notification from the Bridge E-speak Core will thus

be handled and the according action for the respective payload-type taken.

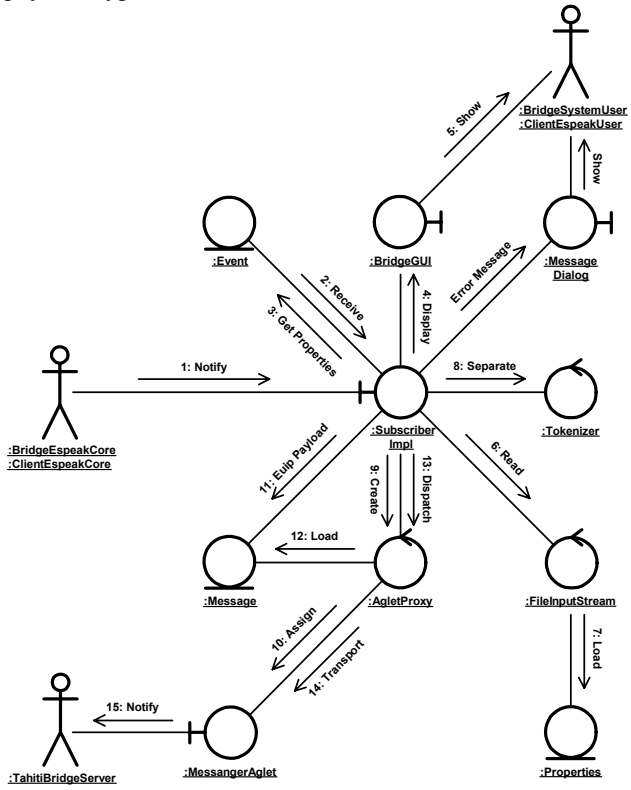


Figure 8: E-speak Event Notification Diagram

5.5 'Release Connection'

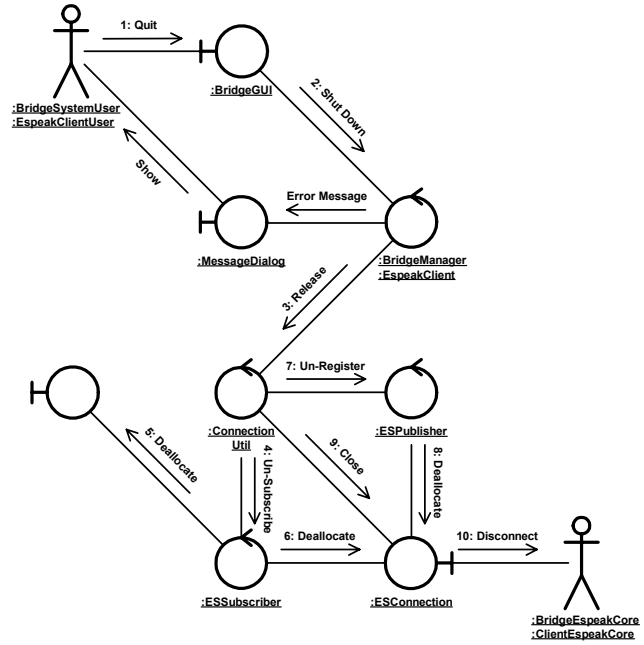


Figure 9: Release Connection Collaboration Diagram

In order to free the resources occupied by E-speak, the E-speak connection can be released. The E-speak Client software or the Bridge Manager software unsubscribe as receiver for E-speak event notifications of the given event type. The E-speak Client software or the Bridge Manager software de-register with the Bridge E-speak Core as the E-speak event publisher for the given event type. Then, the E-speak connection to the Client E-speak Core and from there to the Bridge E-speak Core or respectively directly to the Bridge E-speak Core is closed down. Any additionally allocated resources are also released and all results are verified.

5.6 'Send Aglet Object'

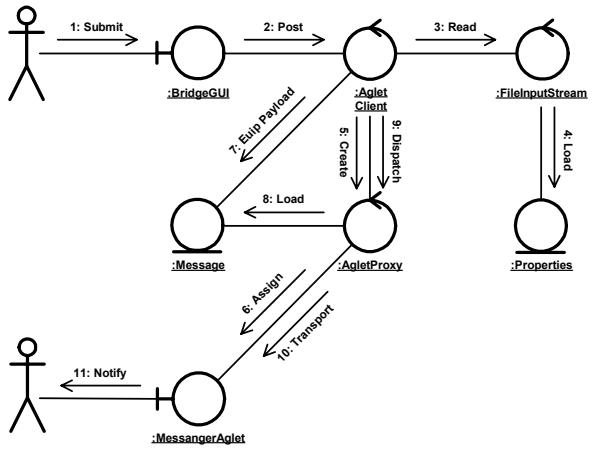


Figure 10: Send Aglet Object Collaboration Diagram

This use case can be invoked after the Aglet Client software has successfully initialized. During start-up, the Aglet Client software automatically subscribes for notifications with the Client Tahiti Server. The Aglet Client User can now send any type of object either by directly accessing the sending method and providing the payload object, or by creating message at the Client GUI and pressing the "Forward" button. The Transport Object use case will be invoked to send the object to the Bridge Tahiti Server, where the Notify Aglet Event use case will take care of the distribution to all Aglet Client Users. Additionally, the Distribute Events use case will perform the distribution to all E-speak Client Users. In this way, the event together with the payload object will be distributed to all Aglet Client Users and all E-speak Client Users. The Client GUI will be updated with the system's status information and the information of the object being sent. Exception handling will be triggered if any operation cannot be performed.

5.7 'Aglet Event Notification'

As before, this use case can be invoked after the Aglet Client software has successfully initialized and subscribed for notifications with the Client Tahiti Server. The Client Tahiti Server invokes the use case, when it receives a Messenger Aglet. The object provided together with the notification message will be extracted and its object-type will be identified. The Receive Object use case will be invoked to handle the respective object-type. That use case will display the received result in the according way inside the Client GUI and will make the object available for external software. The notification will thus be handled and any type of payload-object can be received.

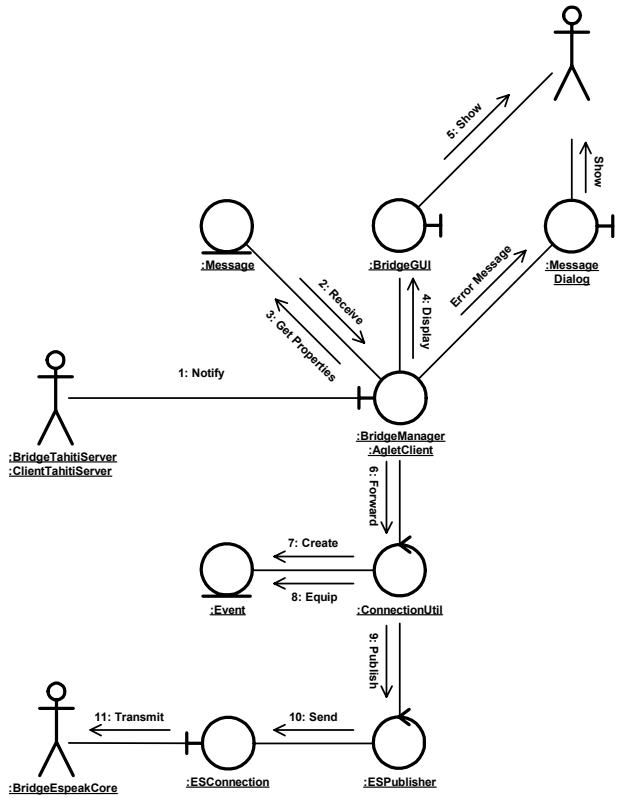


Figure 11: Aglet Event Notification Collaboration Diagram

6 Conclusion and Future Work

A software collection of utility methods necessary to perform collaboration, intercommunication, translation, and bridging between Aglets and E-speak was successfully developed. The provided modules can be used for every existing application written in E-speak or Aglets to create common services together with any number of other Aglets and E-speak applications [10]. The whole software is implemented platform-independently without any hardware-specific properties and designed in a highly modular way.

This paper also points out the benefits of an integration of Aglets and E-speak. Both technologies are examined in terms of how they can support asynchronous, loosely coupled E-Commerce-specific information exchange, bulk data transfer, firewalls, collaboration and dynamic behavior. These comparisons conclude with elaborations on how to overcome persistent problems in E-Business.

Suggestions include how to deploy XML as the common language for information exchange for Aglets, E-speak and the World Wide Web. A combined model for collaboration and mutual assistance between Aglets and E-speak is then suggested. This describes the strengths and domains for each technology and the synergistic benefits of combining their services. E-speak could serve as the communication backbone of an Aglet domain (e.g. an organization) and connect easily through firewalls to any number of other Aglet domains. E-speak would add fine-grained access control, advertising, discovery, composition, metering, billing, event aggregation, and firewall traversal to the combined model.

The software from our project could become the gateway between Aglet domains, E-speak domains and ordinary Web servers. Furthermore, E-speak could take care of all the bulk data transfer to ensure the required throughput and privacy. Aglets could enfold their functionalities within their domain (organization) as autonomous distributed components, offering greater flexibility and acting on behalf of their clients to fulfill many types of tasks.

References

[1] Hewlett Packard (2000) Ten Ways to Think E-speak. *E-speak Documentation*. http://www.e-speak.net /library/pdfs/ThinkEspeak.pdf.

[2] Hewlett Packard (2000) Architecture Specification. *E-speak Documentation*. http://www.e-speak.net /library/pdfs/E-speakArch.pdf.

[3] Hewlett Packard (2000) JESI Programmers Guide. *E-speak Documentation*. http://www.e-speak.net /library/pdfs/Jesi-PgmGuide.pdf.

[4] Hewlett Packard (2000) Service Framework Guide. *E-speak Documentation*. http://www.e-speak.net /library/pdfs/ServiceFramework.pdf.

[5] Hewlett Packard (2000) Contributed Services. *E-speak Documentation*. http://www.e-speak.net /library/pdfs/ContributedServices.pdf.

[6] Hewlett Packard (2000) WebAccess Programmers Guide. *E-speak Documentation*. http://www. e-speak.net/library/pdfs/Webaccess-PgmGuide.pdf.

[7] IBM (2000) Aglets Software Development Kit. *Aglets Documentation*. http://www.trl.ibm.co.jp /aglets.

[8] Lange D B, Oshima M (1998) Programming and Deploying Java Mobile Agents with Aglets. *Addison-Wesley*.

[9] Moukas A, Guttman R, Zacharia G, Maes P (1998) Agent-Mediated Electronic Commerce. *MIT Media Laboratories*. http://ecommerce.media.mit.edu.

[10] Schmid A (2001) Mapping of Aglets into E-speak. *Information Communication Institute Singapore*. Nanyang Technological University Singapore.

An Impact of ICT – Assesment of Indicators on National and Companies' Level

Cene Bavec
University of Primorska, Faculty of Management
Cankarjeva 5, Koper, Slovenia
e-mail: cene.bavec@guest.arnes.si

Maja Bučar
Center of International Relations, University of Ljubljana, Faculty of Social Sciences
Kardeljeva ploščad 5, 1000 Ljubljana, Slovenia
e-mail: maja.bucar@guest.arnes.si

Metka Stare
Center of International Relations, University of Ljubljana, Faculty of Social Sciences
Kardeljeva ploščad 5, 1000 Ljubljana, Slovenia
e-mail: metka.stare@guest.arnes.si

Received: July 10, 2003

Until recently, discussion of the ICT impact has been focused on very few issues, predominantly economic growth and productivity. It is becoming increasingly evident that impact is much wider and affects all spheres of economic and social life. The evidence from OECD member countries is reaffirming the importance of ICT as a driver of growth at all levels: at the level of national economy, at the sectoral level and at the level of individual companies. Recognizing different dimensions of ICT and its interlinkages among different sectors is important to fully asses its potential in national economy and individual company. Many argue that current ICT indicators often deal with less important issues and miss to address "softer" views of digital economy like ICT impact on product or service quality, organizational change, quality of human capital, level of globalization, or government policies. In the paper we discuss and asses statistical indicators used at different levels of decision making – from national to companies' levels. We attempt to verify their relevance and usefulness for various purposes. The case of Slovenia is presented to illustrate interpretations and assess presently used indicators.

1 Introduction

A transition to the information society is constantly monitored by different national and international bodies. Particularly the OECD and the European Commission have introduced many internationally comparable statistical indicators to support high level decision makers. Nevertheless, we are still facing some basic dilemmas:

- Do we use relevant indicators of information society?
- Do we even know what to measure?
- Do we really understand the impact of measured quantities on development of information society?
- Which indicators are relevant for national, sectorial or companies' levels?

Measuring information society and ICT impact on economy and society is obviously a difficult task. National and international statistical systems were developed in a reasonably stable and predictable economic and social environment. Now, we are moving towards fast changing and not fully predictable society. Under new circumstances, many traditional statistical methodologies and even indicators are to be questioned in terms of their value. Development and assessment of information society indicators is a challenging research area. It becomes a multidisciplinary endeavor which has to attract researchers from many scientific fields far beyond technology and economics.

ICT has received an undivided attention as an engine of growth, with several governments counting on this sector as the one making the most important contribution to the economic growth, productivity, employment, and national competitiveness. Investing in ICT is important also at the firm's level, where many protagonists of new technologies were making very optimistic promises of dramatic improvements in business results.

Correct assessment of all potential impacts of ICT is therefore important at the national level, but it is an even more challenging issue for business companies. To sail through unpredictable waters of new economy, management must rely on different information sources to asses their position. One source of valuable information is ICT indicators gathered by national

statistics, but this is far from being enough. Therefore, the companies are forced to collect their own data, and even trust their own good guesses and managerial intuition.

The basic statistical source presented in this paper is recently issued OECD 2003 Report: ICT and Economic Growth – Evidence from OECD countries, Industries and Firms (OECD, 2003). An impact of ICT on the economic growth and productivity at the national level was examined, looking both at the contribution of ICT producing and ICT using sectors. Most interesting are the results of the firm-level studies, where important cross-country differences in firms' use of ICT were found. The analysis of the report suggests at least two things. International research trends must be accompanyed with national (Slovenian) research efforts to be able to make an objective international and sectoral comparisons and interpretations. The second issue is an integration of macro (national) and micro (company) indicators into a meaningful and harmonized set of indicators which could support policy and decision making at different levels.

2 Impact of ICT on National Level

2.1 ICT and Economic Growth

For the OECD countries growth accounting estimates show that ICT investment typically accounted between 0.3 and 0.8 % of growth in GDP over the 1995-2001 period. This justifies ICT investment as an important indicator, regardless of anticipated correlation between GDP and ICT. Figure 1 reveals an interesting and meaningful fact that this correlation is not linear, but increases with higher levels of ICT investment.

In country studies, where researchers use so calles hedonic indexes- accounting for the change in quality of ICT equipment at the same time as falling prices are occuring (see Bucar, 2002 for more detailed explanation) are likely to record even higher ICT contribution to growth performance. The same is happening with the impact on labor productivity of both ICT-production and ICT-using sectors. Variations are larger in measuring the contribution of ICT to productivity in service sectors, where in fact initially the introduction of ICT in some services show low or even negative impact on the growth of productivity. Later studies (Oliner and Sichel, 2002, Triplett and Bosworth, 2002) explained that in service sector one needs to take into consideration also the changes in variety of services provided (for example availability of ATM in banking) due to the ICT use as well as the quality impact.

These findings need to be complemented by the company-level indicators to get better insight into complicated interplay and relations between ICT and its impacts.

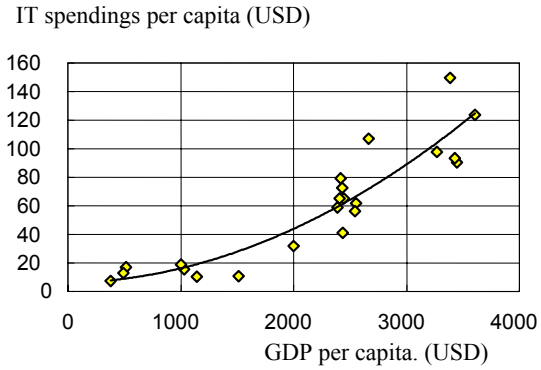


Figure 1: *Correlation between GDP and ICT (presented are EU and EU Accession countries, USA and Japan) (Source: WCY, 2003)*

2.2 Diffusion of ICT

One of the more standard indicators of the ICT diffusion is the share of ICT in investment. As expected, the share of ICT investment was and still is the highest in United States (slightly less than 15% in the eighties, above 20% in the nineties and nearly 30% in 2001). Other countries show similar trends, but at lower levels - United Kingdom (22% in 2001), Sweden (22%) and the Netherlands (21%), EU average (12-17%). Slovenia is the low end of EU, since ICT investment account for 14% of all investment in 2001. (Stare, 2003) Diffusion of ICT is stronger in more competitive environment, confirming the expectations of diffusion theory where the spread of new technologies and innovation depends on the level of market competition: in a more protective environment companies don't need to innovate and constantly upgrade their technologies.

The second determinant of the importance of ICT is the size of the sector that produces ICT goods and services. Having a strong ICT producing sector may help generate the skills and competencies needed to benefit form ICT use. Also important is its contribution to growth, employment, and exports of this very fast growing economic activity. Surprisingly, with exception of few countries (among them Finland, Ireland, Korea, USA), ICT producing sector is relatively small (4-17% of business sector value added, 6-7% of total business employment, but nearly 18% of total trade in 2000 in OECD countries). This confirms earlier findings, that while ICT sector is important, it is not a prerequisite for growth (OECD, 2001).

In Slovenia the ICT sector recorded a dynamic growth in the second part of the nineties and in 2001 accounted for 7.5% of value added and 5% of employment of the total Slovenian non-financial corporate sector. Compared to 1995 when the respective shares in total value added and in employment amounted to 5.3% and 3.4% the importance of ICT sector increased significantly (Figure 2).

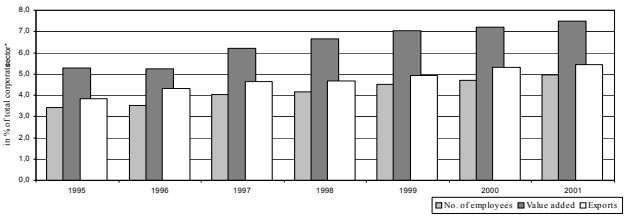


***Figure 2**: Importance of ICT sector in the non-financial corporate sector* in Slovenia, Source: Stare. Kmet, Bučar, 2003*

In the period 1995-2001, the number of ICT sector companies in Slovenia increased from 993 to 1654 and the total number of employees grew from 16,591 to 23,532. The above data show that ICT sector occupies an increasing share of the Slovenian economy, however its weight remains lower than in developed economies or in some transition countries (e.g. in Hungary, Czech Republic) (OECD, 2003). The bulk of the ICT sector growth in the period 1995-2001 resulted from dynamic growth of ICT services

The importance of ICT sector can also be assessed in terms of its productivity. In Slovenia, telecommunication services recorded the highest value added per employee (69,400 EUR in 2001) and thus reached approximately 70% of the respective EU productivity in 1997. Computer services productivity follows, accounting for 66% of the EU productivity level. This is considerably higher compared to overall productivity of Slovenian economy, which stands at 45% of the EU productivity in 2001 (Bešter, Uršič, 2002). Taking into account intensive interlinkages of ICT services with other activities relatively high productivity of ICT sector in Slovenia points to its significant potential to increase the efficiency of the total economy.

In the case of Slovenia, numerous international surveys rank it high or even the highest among the transition countries in terms of telecommunication infrastructure and equipment, fixed and mobile lines, PCs per inhabitant and Internet penetration (Statistics in Focus, 2002, ICT Enlargement Futures, 2002, SIBIS+, 2003). However, Slovenia still lags behind EU countries and reveals weaknesses of less developed and competitive markets.

3 Impact of ICT on Company Level

3.1 ICT in business environment

Positive contribution of ICT to business performance has been confirmed by several studies in different countries. For the United States, Stolarick (1999) found a positive relationship between IT spending and productivity, but one that varied between industries. He also found that low productivity plants sometimes spend more on IT than high productivity plants, trying to compensate for poor performance. Even in the case of micro-enterprises, high ICT endowment meant higher innovation, R&D, training, strong inter-enterprise relations, higher productivity and earnings (De Gregorio, 2002). Bartelsman et al. (1996) used data from a technology use survey in the Netherlands. They found that adoption of advanced technology is associated with higher labor productivity, higher export intensity, and larger size.

Assessing the overall impact of ICT (not the most direct correlations of investment and growth) on an individual firm is still more an art than science. There are still few common methodologies that are applied at the national and international levels to allow for comparability. Most of the firm-level studies have so far been based on selected number of cases or data series constructed by individual research groups. This can limit the applicability of their results, since diversity of business organizations is much larger then diversity of governments. This is reflected also in the diversity of data they need for decision-making process. Business approach is also much more pragmatic then national statistical systems and it heavily depends on type of organization, and even on personal style of management.

3.2 Implementation of ICT and readiness for e-business

One of the indicators of information society is the spred of e-business in economy. In most countries, e-business is still emerging and is not a predominant business practice. In the case of Slovenia, half of the companies use certain forms of e-commerce at a relatively high proportion, even by EU standards. On the other hand, the amount of real e-business is still very low. The number of on-line orders is below 5% of the total number of all orders; on-line generated income is below 1% of all companies' income; and on-line retail is below 0.1% of total retail and is applied only in a small percentage of companies (Vehovar, 2002). Many argue that e-business has not been even lunched yet.

Managers' perception of e-business is very pragmatic. They are aware that digital economy is their future and they are left with no alternative but to invest into ICT. But, they are quite conservative and they implement new technologies and services only when competition forces them to do so. The "followers" business philosophy is maybe the most severe obstacle for faster Slovenian transition to information society. It should be of much more concern than what we can notice in IS strategy at the government level.

Our research surprisingly shows that managers believe that individual companies can develop and utilize ICT and new services relatively fast. Therefore, readiness for ICT implementation doesn't depend very strongly on past ICT investments, it is more a matter of "management's state of the mind" and their ability to seize technological opportunities. Such attitude is confirmed by EITO research (Figure 2) on country levels. It implies that readiness for e-commerce only slightly depends on economic strength of the country.

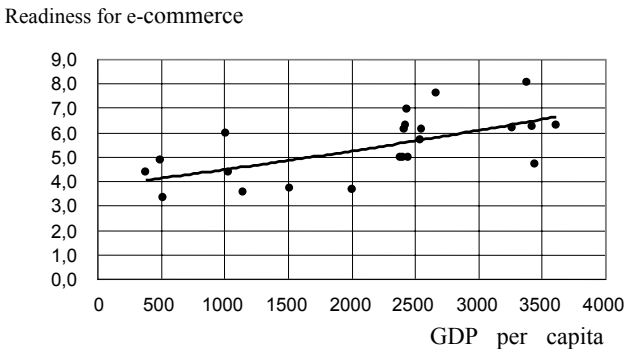


Figure 3: *Correlation between GDP and readiness for e-commerce (presented are EU and EU accession countries, U,S and Japan) in 2000 (Adopted from EITO 2001)*

On the other hand, OECD research (OECD, 2003) points to the fact that the longer ICT is present in certain economy, the more widely-spread is its use and more sophisticated applications are introduced. This poits to a high level of synergies, created by ICT: or to put it simply; if only a small number of firms in a national economy is equipped with e-business solutions, the business impact is likely to be modest in comparison to the environment, where majority uses similar business mode.

Absorption ability of business for new technologies strongly depends on the human factor. It is even more important for development of the e-business, where a mixture of managers and technicians with technical and business backgrounds is crucial. In the Central European region we could find IT professionals, but it is a general assumption that we face a lack of flexible and entrepreneurial managers (Bavec, 2000). Some researches show (WCY 2002) that the situation is not so dramatic in Slovenia, which scores high even among EU countries in the entrepreneurship skills of its managers.

3.3 ICT influence on organizational changes

Another widely discussed issue is ICT impact on organizational structures and processes in companies. It has turned out that this is the most controversial question of all. All research confirms that management often deals with organizational changes on very intuitive way. Researchers and business practitioners all recognize that an appropriate organization is one of the crucial factors of success in digital economy, but there are no statistical indicators that deal with this issue. Theory and practice confirm a transition from rigid traditional organizations towards more organic and even virtual organizational structures, but we still don't have any meaningful and useful indicators to qualify such transition(Bauer and Köszegi 2003, Bavec 2002).

The theory of virtual organization demonstrates that trust among business partners is one of basic requirements for implementation of new organization paradigms (Mowshowitz 1999). It is relatively easy to establish a bilateral trust between two companies, but it is much more difficult to achieve a global trust of an e-market on national or global level. Trust depends on legally imposed mechanisms like electronic signature and legal recognition of electronically signed documents, but it is also based on psychological perception and business culture in certain environments. Interviews with managers in Slovenia reviles surprisingly high level of trust among business partners what makes Slovenia slightly different from other Central European countries. This could turn out to be an important advantage in our transition to digital economy.

3.4 What could affect ICT impact

A common finding in several micro-level studies in OECD countries of the ICT impact was that the positive impact on firms' performance was not only the result of ICT, but of several complementary changes in organisation and skills as well as additional investment in other areas. This simply means that ICT does not not work in isolation and that simply increasing the investment in ICT and expecting improved business results will not do the trick.

Skills

Computer-based technologies are used by workers with higher skills. In several cases, the introduction of ICT increased also the companies' budget for education and training and reduced the employment of unskilled and low-skilled workers (Baldwin et. al, 2002; Falk, 2001; Caroli and Van Reenen, 1999 and others). In several micro-level studies the level of human capital was linked to the rate of diffusion of ICT. This confirms the complementarities between technology and skills in improving productivity performance. In the case of Slovenia, lack of sufficiently skilled workers (not just IT specialists) was ranked number one obstacle to faster ICT diffusion (Bučar, 2001).

Organisational change

Greatest benefits from ICT are realised when ICT investment is combined with other organisational change, such as new strategies, new business processes and practices and new organisational structures. These include teamwork, flatter management structures and employee involvement and suggestions schemes. Close correlation was found of all three categories: skills level, ICT diffusion and organisational change: the companies with highly skilled employees were able to execute more profound organisational change, supported with the introduction of ICT. Just the opposite: ICT uptake with no organisational change (due to policy barriers or internal resistance) did not lead to the expected economic benefits.

Firm size and age

Most of the micro-level studies found that the rate of adoption of advanced technologies, such as ICT,

increases with the size of firms and plants. Yet, small new firms are more likely to adopt ICT than small old firms. The network technologies (Intranet, Internet, EDI) are quite widely used regardless of size, but the purpose of use can be quite different for a larger (internal communication, outsourcing, B2B) than for a smaller firm (marketing).

Ownership, competition and management
The ownership change commonly leads to organisational change and introduction of advanced technologies. Several firm-level studies found that international competition was an important factor in firm's decision to invest in ICT. This would hold true for Slovenian frims as well, since foreign owned firms and those who are integrated in supply network abroad are among the most active users of ICT. Management skills affect the ICT impact, since it is commonly the top management decision to go forward with ICT investment.

Innovation
There is an important link between the use of ICT and the ability of a company to adjust to changing demand and to innovate. This is reflected even in aggregate data: countries that have invested most in ICT also have the largest share of patents in ICT (OECD, 2002). Both correlations were confirmed by research: ICT is an important enabler of innovation, especially in service sector; but also firms that had introduced process innovations were particularly successful in using ICT (Hempell, 2002).

Time
The role of time is significant as well. Productivity effect may only become obvious with some time lag, since it takes time to fully adapt to ICT. The longer the advanced technologies are present in the firm, more complex forms are being used (Clayton and Waldron, 2003

Country differences
Here data is still rather scarce, primarily since data sources were of ad hoc nature and not comparable across countries. The only interesting difference found was that firms in all categories of investment had much stronger productivity growth in the US than in Germany (Haltiwanger et al., 2002). This may be because US firms engage in more experimentation and take greater risks than their German (European) counterparts.

Government policies
Government ICT policies are important tools in improving ICT impact on all sectors of society and economy. Proactive policies and national strategies should go in hands with a common EU strategy related to the information society issues (For example eEurope 2005, eEurope+, etc.). Particularly business sector expects additional government activity that will stimulate ICT industry and innovative services. Research in Slovenia shows (Bavec, 2003) that top management believes that the most important tools in the government's hands are fiscal policy, ICT procurement in public sector, educational system, and raising the general public awareness of the strategic importance of ICT. Politics plays an extremely strong role, as well. In the countries that are successful in transition to IS, politics was able to reach a wide consensus on national priorities and government measures. Countries that are not able to reach such a development consensus are not able to introduce efficient policies to support ICT and assure smooth transitions to the information society.

4 Discussion and Conclusions

Information Society indicators are based on traditional national statistical methodologies and are better suited to industrial than information society. Ongoing research reveals that many driving forces of the new economy are still hidden and consequently not measured. We could also argue that official statistical indicators deal with less relevant but easy obtainable technological figures (like no. of telephone lines, etc.) and miss to assess some "softer" views of digital economy like ICT impact on value added, organizational changes, quality of human capital, level of globalization, or even government policies.

Nevertheless, present indicators show significant changes in our economic and social environment and our departure from an industrial society. We can demonstrate continued positive contribution of the ICT to productivity growth. But at the end of the day, the benefits are likely to be larger in the countries where business environment will enable firms to make smart and effective use from this technology. Important elements of such business environment are exactly the factors affecting the ICT impact: sufficiently high level of human capital, readiness and knowledge of organizational change, competitive markets, and supportive national innovation system.

Still, the decision makers in business often need radically different indicators to asses their position and to produce a reasonable business plans. Their views and challenges are too seldom reflected in existing ICT indicators. Development of a consistent set of indicators covering macro and micro levels is still a real research challenge. Their synergy could deepen our understanding of new economy and information society.

5 References

[1] Baldwin, J.R. and D.Sabourin (2002) Impact of the Adoption of Advanced ICT on Firm Performance in the Canadian Manufacturing Sector, STI Working Paper 2002/1, OECD, Paris.

[2] Bartelsmann, E.J.G.van Leeuwen and H.R. Nieuwenhuijsen (1996), Advanced Manufacturing technology and Firm Performance in the Netherlands, Netherlands official Statistics, Vol.11, Aut., pp. 40-51.

[3] Bauer R., Köszegi S.T. (2003): Measuring the Degree of Virtualization, Electronic Journal of Organizational Virtualness, Vol. 5, No. 2

[4] Bavec C. (2000), On the Information Society in the Central European Countries, World Research Market, London
[5] Bavec C. (2002): An assessment of the Organization Virtuality with three different reference models, Informatica, 26 (2002)
[6] Bavec C. (2003): Indicators of ICT impact on the level of an individual company (in Slovenian), Report on Research Project, Facuty of management, Koper
[7] Bešter J., Uršič S. (2002): Analiza konkurenčnosti po dejavnostih glede na slovensko povprečje in v primerjavi z državami EU: Storitvene dejavnosti (The Analysis of competitiveness of Slovenian service activities compared to the EU). Inštitut za ekonomska raziskovanja.
[8] Bučar M. (2001) Razvojno dohitevanje z informacijsko tehnologijo? (Catching-up with Information Technology), Fakulteta za družbene vede. University of Ljubljana. 2001.
[9] Bučar, M. (2002) The impact of iCT on Growth and Productivity; Conference Proceedings of IS 2002.
[10] Bučar, M. (2003) Economic growth and information Communication tehnology: national, sectoral and firms' level, Conference Proceedings of IS 2003.
[11] Caroli e. and J. Van Reenen (1999) Organisation, Skills and Technology: Evidence from a Panel of British and French Establishments, IFS Working Paper Series W99/23, Institute of Fiscal Studies, Aug.
[12] Clayton, T. and K. Waldron (2003) E-Commerce Adoption and Business Impact, a Progress Report, economic Trends, forthcoming.
[13] De Gregorio, C. (2002), Mocro Enterprises in Italy: Are ICTs and Opportunity for Growth and Competitiveness?, OECD workshop on ICT and Business performance, ISTAT, Rome.
[14] Haltiwanger J., R.Jarmin and T.Schrank (2002), Productivity, Investment in ICT and Market Experimentation: Micro Evidence from Germany and the United States., OECD Workshop on ICT and Business Performance, Dec.
[15] Hempell, T. (2002) Does Experience Matter? Productivity Effects of ICT in the German Service Sector, DP No.02-43, Centre ofr European Economic Research, Mannheim.
[16] Milana C. and A.Zeli (2001), The Contribution of ICT to Production Efficiency in Italy: Firm-Level Evidence using DEA aand Econometric Estimations, STI Working Paper 2002/13, OECD.Paris.
[17] Motohashi, K. (2001) Economic Analysis of Information Network Use: Organisational and Productivity Impacts on Japanese Firms, Research and Statistics Department, METI.
[18] Mowshowitz A. (1999): The Switching Principle in Virtual Organization, Proceedings of the 2nd International VoNet Workshop, September 23-24, 1999, Simowa Verlag, Bern
[19] Murn A, Kmet R. Poročilo o razvoju 2003 (Slovenian Development Report 2003). Institute of Macroeconomic Analysis and Development, Ljubljana. 2003.
[20] OECD (2001) The New Economy: Beyond the Hype, Paris.
[21] OECD (2002) Measuring the Information Economy, Paris; also available at www.oecd.org/sti/measuring-infoeconomy
[22] OECD (2003) ICT and Economic Growth: Evidence from OECD countries, industries and firms, Paris.
[23] Oliner S.D. and D.E.Sichel (2002), Information technology and Productivity: Where Are We Now and Where Are We Going?, Federal Reserve Bank of Atlanta Economic Review, third quarter, pp.1-13
[24] Papows, J. (1999), Enterpise.com, Market Leadership in the Information Age, Nicolas Brealey Publishing, London.
[25] Stare M. (2003), Dimensions and Intelinkages of ICTin Slovenia, Conference Proceedings IS 2003.
[26] Stare M., Kmet R., Bučar M. (2003) Factors and Impacts in the Information Society: A Prospective Analysis in Slovenia. Institute of Macroeconomic Analysis and Development.
[27] Stolarick, K.M. (1999), Are Some Firms Better at IT? Differing Relationshiups between Productivity and IT Spending, CES WP-99-13, Center for Economic Studies, Washington, DC.
[28] Triplett, J.E. (1999), The Solow Productivity Paradox: What Do Computers Do to Productivity, Canadian Journal of Economics, Vol.32, No.2, pp.309-334.
[29] Vehovar, V. (2002): IT in Slovenia, Presentation at IFIP 2002 Conference, Bled
[30] WCY (2002 and 2003): The World Competitiveness Yearbook, IMD, Lausanne

Categorization of Numerical Values for DEX Hierarchical Models

Martin Žnidaršič[1], Marko Bohanec[1,2] and Ivan Bratko[1,3]
[1] Department of Intelligent Systems, Jožef Stefan Institute
Jamova 39, Ljubljana, Slovenia
[2] Faculty of Administration, University of Ljubljana
Gosarjeva 5, Ljubljana, Slovenia
[3] Faculty of Computer and Information Science, University of Ljubljana
Tržaška 25, Ljubljana, Slovenia

Keywords: decision support, multi-attribute decision models, categorization

Received: July 28, 2003

DEX is a multi-attribute decision modelling methodology. Its specialty is the use of ordinal data and qualitative utility functions. Numerical attributes must be therefore categorized before use in DEX models. We present the problem of numerical data categorization and propose two methods which simplify and partly automate this task. The methods suggest interval bounds according to the desired number of categories and the preference curve of attribute values. We implemented both methods and made some experiments with typical inputs. The most interesting results are presented and analysed.

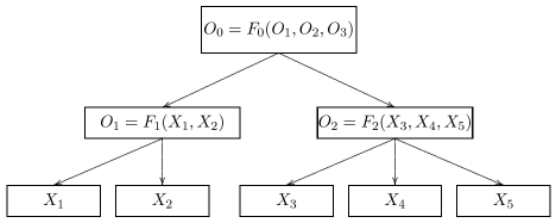


Figure 1: Example of a general MADM.

	Transport	Camping	Price
1	very high	very high	**inappropriate**
2	very high	high	**inappropriate**
3	very high	low	**less appropriate**
4	very high	very low	**appropriate**
5	high	very high	**inappropriate**
6	high	high	**less appropriate**
7	high	low	**appropriate**
8	high	very low	**appropriate**
9	low	very high	**inappropriate**
10	low	high	**less appropriate**
11	low	low	**appropriate**
12	low	very low	**very appropriate**
13	very low	very high	**less appropriate**
14	very low	high	**less appropriate**
15	very low	low	**appropriate**
16	very low	very low	**very appropriate**

Figure 2: Utility function in tabular form defined in application DEXi [6]. The utility of the price for a camping place is evaluated on the base of the transportation cost and the camping fee.

1 Introduction

Data comes in many forms and usually only some of them are useful for a specific task. In this paper, data represents alternatives for multi-attribute decision models (MADM) [8, 7, 4, 9, 1]. With MADM, we are trying to evaluate, compare and study alternatives. Examples of alternatives are for instance cars, job candidates, office locations, etc. Usually we are trying to select the most appropriate alternative for our goals, the one with the highest utility.

Decision problem-solving with MADM is based on hierarchical decomposition of the problem. Alternatives are hierarchically decomposed into subconcepts (or aggregate attributes) and finally to a finite set of basic attributes. Utilities of aggregate attributes are evaluated with functions, which depend on the corresponding attributes located on the lower levels of the hierarchy. A scheme of a general MADM is shown in Figure 1. The MADM tools usually allow the analysis (alternative ranking, sensitivity analysis) and graphical representation of the decision problem.

One of many MADM methodologies is DEX, developed at Jožef Stefan Institute [1, 6]. Unlike traditional methodologies, DEX uses qualitative variables instead of numerical, what makes it suitable for less formalized decision problems. Utility functions in DEX are adjusted to qualitative variables and therefore represented with if-then rules, usually given in tabular form (as in Figure 2). This qualitative approach proved to be very useful in practice, since DEX was used in many real-world decision problems [2, 3].

Numerical attributes often occur in the lower levels of DEX models. In such cases the qualitative approach of DEX is a drawback, as we can not define a rule for every value of a numerical attribute. Sometimes we can describe

a certain feature with categoric, instead of numeric values, but when the use of numeric data is obligatory, the only solution is to define intervals of numerical values and to use such intervals as attribute values. The process of splitting attribute values into intervals is called categorization. Selecting appropriate interval bounds is not a trivial task, as the selection is not always obvious and can have a significant impact on the model. We propose two methods to simplify and partly automate this task.

2 Transforming numerical attribute to ordinal one

Numerical attributes are attributes that have numerical values, either continuous or discrete. We will consider only numerical attributes that have continuous or quasi-continuous (discrete with many values) values. Discrete attributes with a very limited range of values can be considered categorical when used in DEX models. Categorical attributes can be nominal (unordered) or ordinal (ordered).

A numerical attribute can be transformed into an ordinal one with categorization. Values of the numerical attribute are divided into intervals, which are then considered as possible values of the new attribute. The division of continuous values into intervals is an unnatural procedure and is difficult even for skilled experts, especially when they can not find sensible bounds. Simple automatic procedures for interval bounds determination, as equal-width for instance, are inappropriate as they usually increase the difference between a decision model and the corresponding real decision environment.

Manual categorization consists of interval bounds selection and ordering of intervals. When domain experts think there is no sensible way of selecting interval bounds, we believe it is best to think about ordering values in continuous space and express our preference with a continuous preference curve. Categorization should then be carried out by a MADM tool. Preference curve is a continuous curve that has a value of preference in the range [0,1] at each attribute value. Lower values mean that attribute values are considered less preferred, higher values on the other hand are at the values of the attribute that are more preferred. The preference 1 denotes an ideal attribute value, whereas the preference 0 denotes the least preferred attribute value.

Two automatic procedures that, given the preference curve, simplify the process of categorization, are proposed in sections 2.2 and 2.3.

2.1 Manual Categorization

DEX methodology currently offers no help with the transformation of numerical attributes to ordinal. The user has to perform this transformation manually. Numerical values of the attribute have to be divided into intervals that form a range of values for the new ordinal attribute. If the intervals are left unordered, the new attribute is nominal. However, it is prefered to order the intervals to obtain an ordinal attribute.

2.2 Method 1: Linking intervals

The first proposed categorization method needs only two inputs from the user, the preference curve P for attribute values and the desired size of the set of values for the new ordinal attribute $|D_{\text{ord}}|$. The values of numerical attribute are initially divided into many small intervals (default $10 \times |D_{\text{ord}}|$). Mean preference value $avgP$ is computed for each interval. Then the two intervals with the most similar $avgP$ are linked (see sketch 2 on Figure 3) and their $avgP$ is updated with the mean of previous two values. Procedure of linking similar intervals continues until there are only $|D_{\text{ord}}|$ different $avgP$ values. Remaining $avgP$ values are ranked and given values from 1 to $|D_{\text{ord}}|$ where a higher number means a more preferred value. Linked intervals with common bounds (neighbors) are merged together. With this final step, the transformation from a given numerical attribute to an ordinal one is completed.

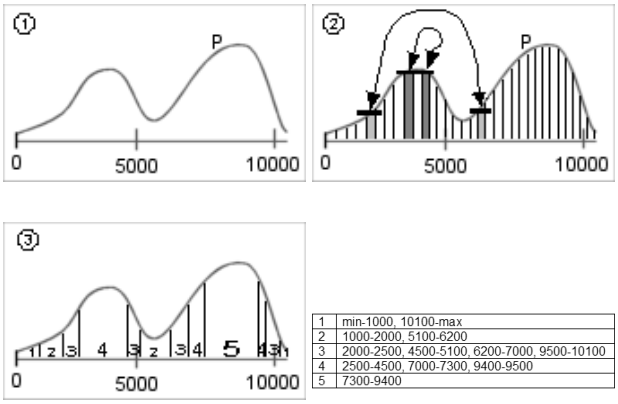


Figure 3: Sketches of the *Linking intervals* method.

Example: transport
A simple explanation of the concepts introduced so far is in Figure 3. Suppose we are in the role of a truck driver. From all the possible transport configurations and routes, we have to choose the most profitable one. To analyze our decision problem, we build a MADM. Basic attributes are for instance: length of transport, price of transport, countries on the way, weight of the load, road fees on the way and similar. Let us analyze the attribute *weight of the load* in more detail. When weight reaches 5000 kilograms, we have to use an additional trailer because of physical limitations and traffic regulations. The optional use of trailer reflects in a bimodal preference curve (see sketch 1 on Figure 3). The most preferred weight of the load is the weight of fully loaded truck and trailer, and a minor local maximum is at the point where the truck without trailer is fully

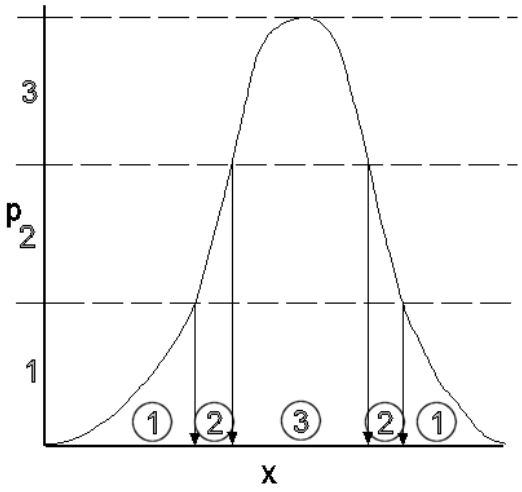


Figure 4: *Following preference* method on Gaussian preference curve.

loaded. Truck with almost empty trailer is not preferred and almost empty truck even less. On the base of such considerations we can construct a preference curve that is used by the method *Linking intervals* to select interval bounds and ordinal values (sketch 3 on Figure 3).

2.3 Method 2: Following preference

Our second proposed method is a simple and computationally less demanding procedure. The preference is divided into $|D_{\text{ord}}|$ equal size intervals that represent the set of ordinal values. Each of these intervals transforms the corresponding attribute values into its ordinal value. The procedure is presented in Figure 4.

It might seem that both proposed methods give identical results, but there are some important differences. *Following preference* method sets the ordinal values strictly regarding the preference, whereas *Linking intervals* method takes the preference into account only to a certain extent. It is more "fuzzy", what makes it less sensitive to sudden changes in preference curve, as well as a bit less accurate regarding preference. Each method has some advantages and some drawbacks, therefore the choice of the right method for a specific preference curve could be important.

3 Implementation and Experiments

Methods *Following preference* and *Linking intervals* were implemented in Delphi environment. Application ACNA for experimenting with the methods is presented in section 3.1. The analysis of methods properties and results is presented in section 3.2.

3.1 ACNA

ACNA (Automatic Categorization of Numeric Attributes) is an application for experimental evaluation and use of the proposed methods for automatic categorization. Graphical user interface facilitates an easy input of parameters and a quick presentation of results in textual and graphical form.

The desired power of the set of values for ordinal attribute and preference curve must be provided. Preference curve must be given as (x, y) coordinate pairs and can be read from file. When entered, preference curve is presented with coordinates and a graph that can be saved as a file.

When all input data is provided, we can start each of the proposed methods. The resulting intervals and their ordinal values appear together with a graph of preference curve divided into calculated intervals. This graph is very useful for a quick overview of results and can also be saved as a file.

3.2 Comparison of Methods

Both proposed methods were tested on four distinctive and very different preference curves. We tested categorization into 2, 3, 4, 5, 6, 7 and 12 values with each curve. We were expecting the method *Following preference* to suggest mostly appropriate divisions. The method *Linking intervals* was expected to be somewhat inaccurate, but we were hoping that it will find a more suitable division in some unusual situations. Of all the experiments we selected only the most interesting ones for the presentation of differences.

Given the preference curve 'increasing', the results of both methods were very similar. An example of categorization into three values is shown in Figure 5. Differences given this type of preference curve were not expected.

Some differences appear when using preference curve 'normal', that has a shape of a Gaussian function. Given this type of preference, regardless of the number of desired categoric values, the method to use is *Following preference* (Figure 6). It is a simple problem that calls for simple solution procedure. Any variations from results of *Following preference* are unwanted. Slight inaccuracy of *Linking intervals* method is reflected in suboptimal selection of intervals.

At a bit more stirred preference curve 'bimodal' reveales an interesting difference in results of the two methods. Results visually do not differ much, but are very different with regard to the number of intervals used in each solution. *Following preference* method proposed for instance 11 intervals for categorization to 5 values. In the same setting, *Linking intervals* method provides a solution with only 7 intervals (Figure 7). A similar trend can be noticed using any other number of categoric values. This effect is a consequence of rigidity of the method *Following preference* and would be even more obvious if the preference curve was noisy (automatic generation) or had a particularly unsuitable shape.

Preference curve that emphasizes stiffness of *Following preference* method is 'stairs' (Figure 8). In some cases of

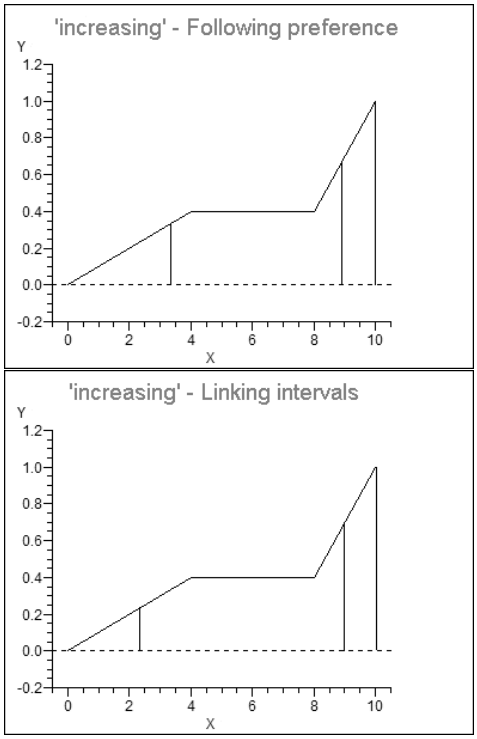


Figure 5: Results of categorization to three values given preference curve ‘increasing’.

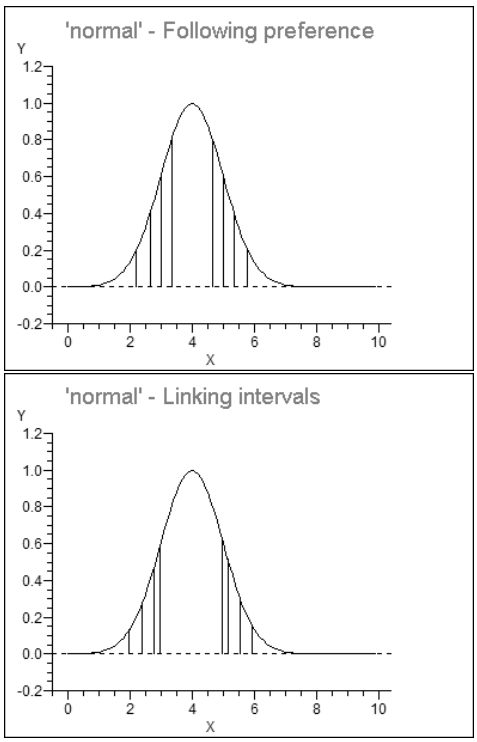


Figure 6: Results of categorization to five values given preference curve ‘normal’.

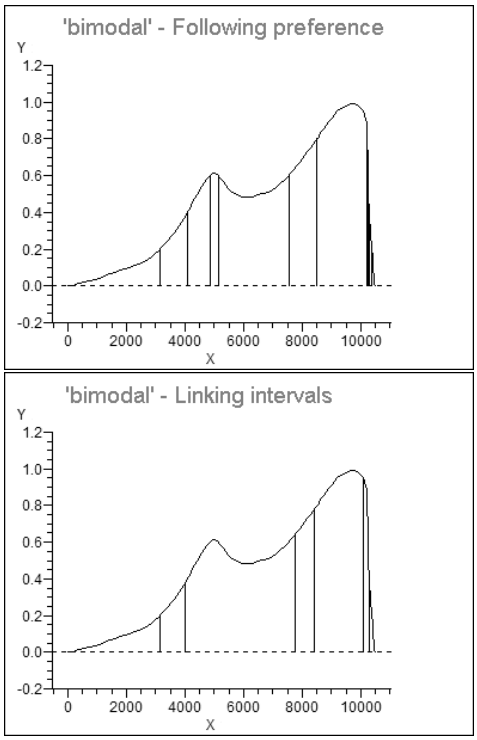


Figure 7: Results of categorization to five values given preference curve ‘bimodal’.

desired number of categorical values, *Following preference* misses the natural course of preference curve. In addition to that, it suggests many unnecessary small intervals when preference curve suddenly drops. In such cases of preference curve we should use *Linking intervals*, that adapts itself to the shape of preference curve.

Generally the *Following preference* method provides suitable results. Proposed solutions of the *Linking intervals* method are usually less appropriate and for use with preference curves similar to Gaussian curve, not appropriate. However, in some cases of uncommon preference curves, *Linking intervals* provides a more natural and simple solution. None of the two methods is best in every situation, so both should be used with care.

4 Conclusion

We introduced two methods to simplify the categorization of numerical attributes in DEX methodology models. The main advantage of presented approach is providing the ability to present data and knowledge in a natural way, suitable for continuous data.

Experiments indicated some interesting features of the methods. Results of the *Following preference* method are generally better, but in certain cases its stiffness demonstrates in unnatural selection of interval bounds. In that situations the more flexible *Linking intervals* method usually gives more appropriate results.

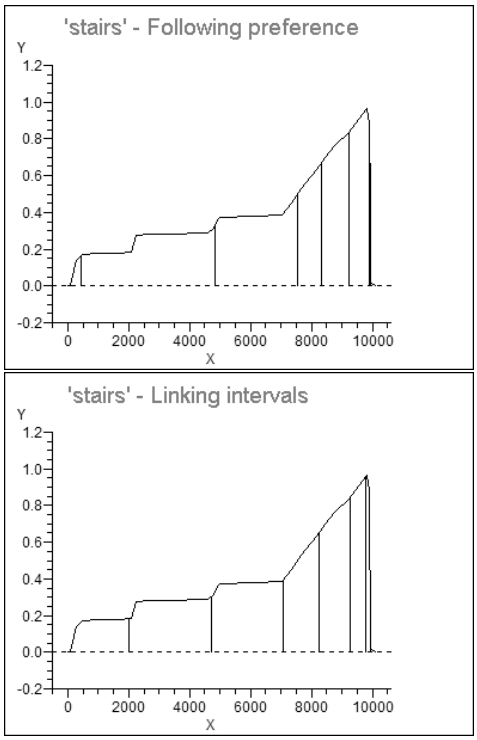


Figure 8: Results of categorization to six values given preference curve 'stairs'.

Method features we discovered will have to be confirmed in practice. Practical experiences and further work are necessary to properly evaluate the applicability of proposed methods. Further work will be focused on development of a method that combines the good features of both presented methods and on study of situations where results of one of the methods prevail. Derivatives could be for instance used to detect changes in course of preference curves. As a minor improvement we plan to allow the input of preference curve in form of a mathematical expression. It would also be interesting to test the methods with automatically acquired preference curves.

References

[1] M. Bohanec and V. Rajkovič. DEX: An expert system shell for decision support. *Sistemica*, 1(1):145–157, 1990.

[2] M. Bohanec and V. Rajkovič. Multi-Attribute Decision Modeling: Industrial Applications of DEX. *Informatica*, 23(4):487–491, 1999.

[3] M. Bohanec, V. Rajkovič, and B. Cestnik. Five decision support applications. In D. Mladenić, N. Lavrač, M. Bohanec, and S. Moyle, editors, *Data Mining and decision support : integration and collaboration, (The Kluwer international series in engineering and computer science, SECS 745)*, pages 177–189. Kluwer Academic Publishers, Boston; Dordrecht; London, 2003.

[4] R. T. Clemen. *Making Hard Decisions: an Introduction to Decision Analysis*. Wadsworth Publishing Company, 1996.

[5] V. J. Easton and J. H. McColl. Statistics glossary - presenting data. www.cas.lancs.ac.uk/glossary_v1.1/presdata.html.

[6] E. Jereb, M. Bohanec, and V. Rajkovič. *DEXi-Računalniški program za večparametrsko odločanje (DEXi-Computer Program for Multi-Attribute Decision Making)*. Moderna organizacija, Kranj, SI, 2003.

[7] R. L. Keeney, H. Raiffa, and R. Meyer. *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. Cambridge Univ Press, 1993.

[8] T. L. Saaty. *The Analytic Hierarchy Process*. McGraw-Hill, 1980.

[9] E. Turban and J. E. Aronson. *Decision Support Systems and Intelligent Systems*. Prentice Hall, 2001.

Evolutionary Optimization of Markers in Clothes Production

Bogdan Filipič
Department of Intelligent Systems
Jožef Stefan Institute
Jamova 39, SI-1000 Ljubljana, Slovenia
E-mail: bogdan.filipic@ijs.si

Iztok Fister
Mura, European Fashion Design
Plese 2, SI-9000 Murska Sobota, Slovenia
E-mail: iztok.fister@mura.si

Marjan Mernik
Faculty of Electrical Engineering and Computer Science
University of Maribor
Smetanova 17, SI-2000 Maribor, Slovenia
E-mail: marjan.mernik@uni-mb.si

Keywords: marker optimization, clothes production, knapsack problem, evolutionary algorithm, empirical study

Received: July 29, 2003

Optimization of markers plays an important role in preparation of order-based industrial production of clothes. Given a matrix of pieces in size numbers and designs, the task is to find a list of combinations of size numbers to accomplish a work order. The outcome of this step influences the number of cut out pieces, the amount of material used in the production phase, and the speed of the work order processing. As numerous factors affect the production costs and several conflicting criteria can be involved in marker assessment, marker optimization is a demanding task. In this paper, minimum number of markers per work order is used as an optimization criterion. Marker optimization is formally defined as a knapsack problem, and an evolutionary algorithm is proposed to solve the task. It is tested on real problem instances from industrial clothes production and compared with several other algorithms. Its results on complex work orders are shown to be superior to those of other tested algorithms.

1 Introduction

The preparatory process for order-based production of clothes consists of four phases: creation, construction, multiplication and combining of markers. In the creation phase, a fashion designer conceives the sketch of a model together with the appropriate materials and designs to be used in production. In the construction phase, a constructor sets the sketch in the basic size and defines the constituent parts, such as sleeves and pockets, and materials, such as lining and buttons. The multiplication phase depends on the sales department which, according to customers' requirements, determines the so-called work order. The work order is a matrix of pieces in size numbers and designs. Once the work order is known, the basic size of the model can be multiplied into additional sizes with respect to the work order. Finally, combining of markers can start where the size numbers should be combined according to a prescription defining the outlook of markers. Depending on the outlook of markers are the number of possible cut out pieces, the amount of material used in the production phase, and the speed of the work order processing, in particular laying and cutting.

The optimization of markers comes before the phase of their combining. The key question for the optimization procedure is what the optimal combination of markers actually is. Is it the lowest number of markers to accomplish the work order, or the combination that results in the shortest time to fulfil the work order, or would it not be even easier to solve the work order with separate markers in one size only? As the factors influencing the production costs are numerous and several conflicting criteria can be involved in marker assessment, the optimization of markers is a demanding and challenging task.

In the paper, the optimization of markers for clothes production is first defined as the knapsack problem. Although the task is multi-objective, a single objective is treated which is usually considered as the most relevant in practice. This is to minimize the number of markers needed for the work order. An evolutionary algorithm with several

variants is proposed for solving the marker optimization task and tested on real-world problem instances. Its results are compared with those produced by other algorithms, and directions for further improvement of the evolutionary algorithm for marker optimization are given.

2 Problem definition

Suppose we have a set of n objects and a knapsack of capacity M. In addition, weights of the objects are given by a vector $W = (w_1, ..., w_n)$, and their profits by $P = (p_1, ..., p_n)$. The task is to find a binary vector $X = (x_1, ..., x_n)$ such that

$$\sum_{i=1}^{n} x_i w_i \leq M \tag{1}$$

and the objective function

$$f(X) = \sum_{i=1}^{n} x_i p_i \tag{2}$$

returns the maximum value. In other words, the objects to fill up the knapsack should be chosen in such a way that the capacity constraint is satisfied and the profit maximized. The knapsack problem is known to be NP-hard [3]. For solving this problem in practice, various approximation and stochastic algorithms are used.

The marker optimization problem in clothes production can be formally treated as a knapsack problem. Here, a work order is given in the form of a matrix A with elements a_{ij} and dimension $n \times m$, where n is the number of sizes and m the number of designs of clothes to be produced. Elements a_{ij} are integer values representing the number of pieces of each size and design. The sizes correspond to the objects in the knapsack problem, and the weights are

$$w_i \in [lb..ub] + \{0\}, \quad i = 1..n \tag{3}$$

where lb and ub are the minimum and maximum number of the same sizes in a marker, respectively. The maximum number of sizes in a marker n_M corresponds to the knapsack capacity.

The binary vector X denotes the presence of the sizes in a marker. The profits are obtained as

$$p_i = \frac{w_i}{s_i} \sum_{j=1}^{m} b_j \tag{4}$$

where s_i is the sum of pieces of the i-th size over all designs in the work order A:

$$s_i = \sum_{j=1}^{m} a_{ij} \tag{5}$$

and $B = (b_1, ..., b_m)$ is a vector of layers for the application of a solution (marker) $y(X) = XW$ to the work order A. The elements of the vector B are obtained by

$$b_j = \min(\frac{a_{ij}}{w_i}), \quad i = 1..n \wedge x_i \neq 0, \quad j = 1..m \tag{6}$$

From Equation (4) it follows that the vectors P and W are highly correlated. A valid solution of the task is any subset $Y \subseteq X$ for which Equation (1) holds. The goal is to find the optimal solution Y^* for which the value of the objective function $f(Y^*)$ is maximum [8].

The objective is therefore to maximize the number of pieces per marker. As a marker only partially solves the work order, a number of markers has to be found to accomplish the given work order. To find an optimal marker at each stage, an instance of the knapsack problem has to be solved, and the assumption is this will result in the minimum number of markers to complete the work order.

3 An evolutionary algorithm for marker optimization

An evolutionary algorithm (EA) can be used to optimize the markers. EAs are stochastic search and optimization algorithms from the field of evolutionary computation [1] which considers biological evolution as an inspiration for computer problem solving. The key idea is to search for god solutions by means of computer-simulated evolution where candidate solutions compete against each other. Their quality in solving the problem is used as a fitness measure. Low-quality solutions are excluded from the process, while high-quality solutions produce offspring and undergo variation. This procedure runs in iterations (see Fig. 1) until a termination criterion, such as a prescribed number of iterations, is fulfilled.

```
procedure Evolutionary_algorithm;
  begin
    t := 0;
    initialize_population P(t);
    evaluate P(t);
    repeat
      t := t + 1;
      select P(t) from P(t − 1);
      variate P(t);
      evaluate P(t);
    until termination_criterion
  end;
```

Figure 1: A sketch of an evolutionary algorithm

Despite the lack of theoretical predictions for the quality of the evolved solutions, EAs are more and more often used for practical problem solving in numerical optimization, production scheduling, complex system design and other domains. Their popularity arises from the simplicity and generality of the algorithms and their ability to find near-optimal solutions. To apply an EA to a particular problem, one has to adjust the problem-specific elements of the algorithm: representation of candidate solutions, initialization of the starting population, fitness function to evaluate the

solutions, the operators to variate the solutions, and values of the algorithm parameters.

In the EA for marker optimization in clothes production, a candidate solution $y(X) = XW$ is represented as a vector of n integers, where n is the number of sizes in a work order. It is obtained as a product of a binary vector X and vector of weights W. In the binary vector, $x_i = 0$ denotes that i-th size is not present in a solution, and $x_i = 1$ denotes the presence of the size. The weights in W are generated in the range (3). The starting population of solutions is created randomly with uniform distribution. The objective is to search for the maximum number of pieces to be obtained by a marker on a work order of the size $n \times m$, hence Equation (2) is used to define the fitness of candidate solutions.

The algorithm includes fitness-proportional selection and two traditional genetic operators to variate the solutions during the evolutionary search process: simple (single-point) crossover and uniform bit mutation [4]. The algorithm parameter values set for experimental runs include population size, number of generations and probabilities of crossover and mutation.

Two approaches to solving the knapsack problem were applied in marker optimization: an EA with penalty function and a repair algorithm [7].

3.1 An evolutionary algorithm with penalty function

Penalty functions are a way of dealing with invalid solutions in EAs. The idea is to impose selection pressure to invalid solutions by assigning them lower fitness. The approach is expected to gradually lead to valid solutions in the population and then search for the best among them. The fitness function in this approach is determined by subtracting the penalty term from the objective function (2):

$$g(Y) = \sum_{i=1}^{n} x_i p_i - \mathrm{Pen}(Y) \tag{7}$$

where p_i is obtained from Equation (4), and $\mathrm{Pen}(Y) = 0$ for valid solutions Y. The number of sizes in a marker for a given problem instance is equal to the maximum number of sizes n_M prescribed in advance.

Penalty function for invalid solutions can be defined in various ways. In our problem, invalid solutions are those with the number of sizes in a marker less than n_M. We use three types of penalty functions with different growth of penalty for violations, i.e. logarithmic, linear and quadratic:

$$\mathrm{Pen}(Y) = \log\left(1 + \rho\,(\sum_{i=1}^{n} x_i w_i - n_M)\right) \tag{8}$$

$$\mathrm{Pen}(Y) = \rho\,(\sum_{i=1}^{n} x_i w_i - n_M) \tag{9}$$

$$\mathrm{Pen}(Y) = \left(\rho\,(\sum_{i=1}^{n} x_i w_i - n_M)\right)^2 \tag{10}$$

where in all cases ρ is

$$\rho = \max_{i=1\ldots n}\left(\frac{x_i}{s_i}\sum_{j=1}^{m} b_j\right) \tag{11}$$

which represents the ratio between profit and weight $\frac{p_i}{w_i}$.

3.2 A repair algorithm

In this approach only the solutions with n_M sizes are evaluated using Equation (2) as a fitness function. However, if a solution is not valid, it is first repaired and then evaluated. Three approaches to repairing the generated vectors are used: heuristic, greedy and random.

Heuristic repair relies on Cauchy-Schwarz inequality [6] for determining the angle θ between two vectors $u, v \in R^n$:

$$\cos\theta = \frac{uv}{\|u\| \cdot \|v\|} \tag{12}$$

Vectors u and v are defined as

$$u = (a_{i1}, ..., a_{im}), \quad i = 1..n \tag{13}$$

and

$$v = \left(\sum_{i=1}^{n}\frac{a_{i1}}{n}, ..., \sum_{i=1}^{n}\frac{a_{im}}{n}\right) \tag{14}$$

and a vector similarity relation, denoted by $\prec$, is defined as

$$u \prec v \;\Rightarrow\; \cos\theta > \cos\vartheta \tag{15}$$

where $\theta = \angle(u, v)$ and $\vartheta = \angle(w, v)$. Relation (15) defines the heuristic order of putting the objects into the knapsack.

The order of picking the objects for the greedy method is defined by

$$\frac{s_i}{\sum_{j=1}^{n} s_j} < \frac{s_{i+1}}{\sum_{j=1}^{n} s_j} \tag{16}$$

and the random method generates the sizes that appear in a solution randomly.

Candidate solutions can either be valid, underestimated or overestimated. Valid solutions need no repair and can be evaluated according to objective function (2). In underestimated solutions the sum of weights is less than the maximum number of sizes in the marker n_M. They are repaired to get valid by inserting additional sizes. This is done either by random generation of weights in the range $[lb..ub]$ or according to relation (15) or (16). In overestimated solutions, the sum of weights exceeds n_M. In such cases randomly selected sizes are excluded from the solutions.

4 Experiments and results

The evolutionary algorithm for marker optimization was tested on ten work orders from industrial clothes production taken from [2]. The orders differ in complexity and their characteristics are summarized in Table 1.

Table 1: Characteristics of the real-world work orders used in testing: n denotes the number of sizes in a work order, m the number of designs, n_M maximum number of sizes in a marker, lb minimum number of the same sizes in a marker, ub maximum number of the same sizes in a marker, h_lb minimum number of layers, h_ub maximum number of layers, and k the total number of pieces in a work order

No.	n	m	n_M	lb	ub	h_lb	h_ub	k
1	5	2	4	1	2	4	50	182
2	9	7	8	1	2	5	40	339
3	9	4	14	1	10	5	40	244
4	6	6	4	1	2	17	70	637
5	6	4	4	1	2	4	50	49
6	8	20	4	1	2	1	60	416
7	29	12	2	1	2	10	40	125
8	23	4	2	1	1	10	40	318
9	20	4	2	1	2	10	40	205
10	45	76	4	1	2	4	60	1236

The objective of marker optimization is to maximize the number of cut out pieces for each marker and consequentially minimize the number of markers needed to accomplish a work order. As a stochastic technique EAs generally return different solutions in multiple runs, hence their results have to be analyzed statistically to check for repeatability. The EA for each problem was executed ten times, and best, worst and average results recorded. The algorithm parameters were set as follows: population size 20, crossover probability 0.8, mutation probability 0.05, and the number of generations 50 for smaller work orders (No. 1–5 in Table 1) and 100 for larger orders (No. 6–10).

An example of results obtained in solving work order No. 6 by the EA with penalty functions is shown in Table 2. It can be seen that the algorithm variant with the logarithmic penalty function is able to find the best result in a single run and on average, and also the worst result in a single run. On the other hand, linear penalty yields much less dispersed results with lower average value. This performance is typical for most work orders. It is to be noted, however, that the EA with penalties was unable to solve large work orders (No. 7–10). An analysis of the population showed that the algorithm was dealing with invalid solutions where the number of sizes in markers was larger than n_M and the fitness values negative. Reducing the degree of violation and then finding valid solutions only worked on smaller work orders.

Table 2: Number of markers for work order No. 6 found by the evolutionary algorithm with penalty functions

Penalty method	best	worst	average
logarithmic	10	20	13.8
linear	14	18	15.6
quadratic	11	19	15.4

Difficulties with penalty functions can be avoided by applying the repair algorithm that maintains only valid solutions at each step of the evolutionary search. It turns out that this algorithm can solve all ten work orders. Results presented in Table 3 for the work order No. 10 illustrate a typical outcome on larger work orders. Greedy selection of sizes is better than the random approach, while the heuristic selection of sizes gives the best average and individual result for the number of markers needed.

Table 3: Number of markers for work order No. 10 found by the repair algorithm with different approaches to size selection

Size selection	best	worst	average
heuristic	59	64	60.8
greedy	60	69	64.1
random	62	69	66.1

In addition to comparing various EAs on real problems from industrial practice, a comparative study with other marker optimization algorithms was performed. The following four algorithms were tested:

- exhaustive search that checks for all possible solutions of a marker,
- a deterministic algorithm with heuristic selection of sizes according to the vector similarity relation (15),
- an approximation algorithm with greedy selection of sizes [5],
- an evolutionary algorithm with solution repairing and heuristic selection of sizes.

Exhaustive search could only be applied on smaller work orders No. 1–5, while for larger orders it would need more space and time. The deterministic algorithm is currently used for marker optimization in the textile factory that provided the test problems for this study.

The results in terms of the total number of markers to solve the work orders No. 1–5 are shown in Table 4. The results for larger work orders No. 6–10 are given in Table 5.

Table 4: Total number of markers for work orders No. 1–5 found by different optimization algorithms

Algorithm	best	worst	average
exhaustive search	44	44	44.0
deterministic	53	53	53.0
approximation	43	51	46.3
EA (repair)	43	50	46.2

To appropriately interpret these results, one should bear in mind that the algorithms were run for each marker to partially solve a given work order. The resulting numbers of

Table 5: Total number of markers for work orders No. 5–10 found by different optimization algorithms

Algorithm	best	worst	average
deterministic	161	161	161.0
approximation	157	163	160.0
EA (repair)	150	161	153.1

markers confirm that solving the problem with maximum number of pieces at each step does not lead to the optimal solution for the entire work order. Regarding the performance of the algorithms no clear conclusion can be made for small work orders, while for more complex ones the EA outperforms other tested algorithms. It therefore seems to be an appropriate candidate to replace the currently used deterministic algorithm.

5 Conclusion

Optimization of markers is a preparatory step for order-based clothes production that critically affects production costs. It can be considered from the point of view of various criteria and remains a challenging optimization task. A specific problem of finding the minimum number of markers to accomplish a given work order can be defined as the knapsack problem, and the algorithms for this problem used to maximize the number of pieces with each marker and consequently minimize the number of markers.

The key contribution of our work is the implementation of several variants of the EA for marker optimization, its application to real-world problem instances from industrial practice and comparison of the results by various optimization algorithms. The numerical experiments indicate the EA with solution repairing and heuristic selection of sizes outperforms other algorithms on complex work orders.

Future work on this problem will include experimental verification of the algorithms on more complex work orders, application of various optimization criteria, either in the form of a weighted sum or by means of search for Pareto-optimal sets of solutions. Finally, a database of the existing markers from previous orders will be utilized in solving new orders and the results compared with the current approach where all the markers are built in the optimization process.

References

[1] Bäck, T., Fogel, D. B., Michalewicz, Z. (Eds.) (1997). *Handbook of Evolutionary Computing*, Institute of Physics Publishing, Bristol, Philadelphia, and Oxford University Press, New York, Oxford.

[2] Fister, I. (2003). *Optimization of markers in clothing industry*, Technical report, University of Maribor, Faculty of Electrical Engineering and Computer Science, Maribor (in Slovenian).

[3] Garey, M. R., Johnson D. S. (1979). *Computers and Intractability, A Guide to the Theory of NP-completeness*. W. H. Freeman, New York.

[4] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA.

[5] Horowitz, E., Sahni, S. (1978). *Fundamentals of Computer Algorithms*, Computer Science Press, Rockville, MD.

[6] Lipschutz, S. (1974). *Theory and Problems of Linear Algebra*, Schaum's Outline Series, McGraw-Hill, London.

[7] Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*, Springer Verlag, Berlin.

[8] Robič, B. (2002). *Approximation algorithms*, University of Ljubljana, Faculty of Computer and Information Science, Ljubljana (in Slovenian).

Increasing Fault-Tolerance of Multi-Agent Systems

Andraz Bezek, Matjaz Gams
Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
{andraz.bezek, matjaz.gams}@ijs.si

Keywords: fault-tolerance, multi-agent systems

Received: June 15, 2003

We analyze fault tolerance in load balancing systems. First, we theoretically analyze some aspects of fault-tolerance of traditional and multi-agent approaches. Second, we investigate efficient fault detection showing that multi-agent systems can increase fault-tolerance by improving fault detection of failed single agents. The presented ideas are applied in a multi-agent system for fault-tolerant network load balancing. A detailed description of fault-tolerant issues in design is also given. Finally, the paper presents evaluation of a multi-agent application for network load balancing.

1 Introduction

Designing and building high quality industrial-strength software is difficult, and it has been claimed that such development projects range among the most complex construction tasks. Among software projects, building reliable large Internet services is one of very difficult tasks especially because virtually continuous uptime and consistent response time are crucial. Stable services must be able to cope with many undesired factors such as explosive growth of traffic over the Internet, and possible hardware and software failures.

A wide range of software engineering paradigms has been devised, and each successive development claims either to make the engineering process easier or to extend the complexity of applications that can feasibly be built. Although there is reasonable evidence to support these claims, researchers continually strive for more efficient and powerful software engineering techniques. Recently, multi-agent systems received a lot of attention as a potential mainstream initiative for distributed software engineering [5]. A multi-agent system (MAS) is a loosely coupled network of software entities that work together to solve problems that are beyond the individual capabilities or knowledge of each entity [2]. Recently, the term MAS has been given a more general meaning, and it is now used for all types of systems composed of multiple autonomous agents showing the following characteristics:

- each agent has incomplete capabilities to solve a problem,
- there is no global system control,
- data is decentralized and
- computation is asynchronous.

MAS is a distributed reactive system, which maintains an ongoing interaction with environment. It has long been recognized that reactive systems are among the most complex types of systems to design and implement [9]. Great effort has been devoted to developing software tools, programming languages, and methodologies for managing this complexity – with some success. But for certain types of reactive systems, even specialized software engineering techniques and tools fail. According to Jennings et al. [4], one can broadly subdivide these systems into three classes:

- open systems,
- complex systems and
- ubiquitous computing systems.

An *open system* is capable of dynamically changing its own structure. Consequently its components can be heterogeneous and added dynamically. They also can be changed over time using different software tools and techniques.

Development of *complex systems* requires efficient tools for handling software complexity. The most powerful concepts are modularity and abstraction. Namely, if a problem domain is particularly complex, large, or unpredictable, then it may be reasonable to develop a number of distinct modular components. An agent-oriented approach is a powerful alternative for making systems modular. In case any interdependent problems arise the agents in the systems must cooperate with one another to ensure that interdependencies are properly managed. In such domains an agent-based approach means that the overall problem can be partitioned into a number of smaller and simpler components, which are easier to develop and maintain, and which are specialized at solving the constituent sub-problems. This decomposition allows each agent to employ the most appropriate paradigm for solving its particular problem, rather than being forced to adopt a common uniform approach that represents a compromise for the entire system, which is not optimal for any of its subparts. The notion of an autonomous agent also provides a useful abstraction in the same way that procedures, abstract data types, and objects provide abstractions.

Ubiquitous systems are expected to ease the interaction between humans and computers. But today the user of a software product typically has to describe each step that needs to be performed to solve a problem down to the smallest level of detail. Ubiquitous system should be able to recognize opportunities and act in such a way to help the users to achieve their goals. One can think of such

systems as assistant agents, capable of acting with the user in order to achieve the user's goals.
In MAS the locus of control is dispersed among agents which coordinate and plan their actions according to their local perception of environment. If we try to improve fault-tolerance of a MAS, we must improve fault-tolerance of all agents within MAS, or provide system-wide methods to enable fault detection and recovery. This paper addresses the latter problem. It overviews fault-tolerance with agent systems and presents a fault-detection models together with a fault-tolerant agent application for network load balancing.
The structure of this paper is as follows: In Section 2, we present functional characteristics and differences between traditional and multi-agent load balancing system. Section 3 deals with fault tolerance in agent societies. An analysis of fault-tolerance related differences for two load balancing systems is presented in Section 3.1. Fault detection models are presented in 3.2 and survey of related work is given in Section 3.3. A fault tolerant multi-agent system is presented in Section 4. Finally, conclusions are presented in Section 5 by reviewing the ideas presented in the paper.

2 Traditional and Multi-Agent Load Balancing

We refer to the term *load balancing* (LB for short) as network load balancing or - more precisely - as IP-level load balancing. The term cluster refers to all computers within a load balancing system, thus including all servers, load balancers and administration computers. Sometimes we refer to a cluster of servers and a cluster of load balancers, each describing a set of computers within the cluster with server/load balancing functionality.
The purpose of network LB system is to evenly distribute incoming network traffic among servers in a cluster. The distribution is done according to desired LB policy, which often takes into account performance metrics such as network traffic or processor load.

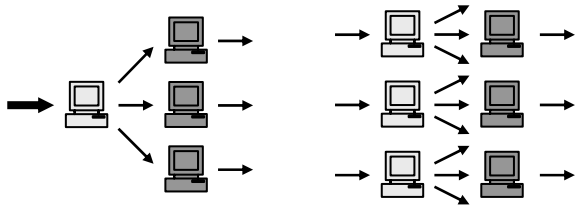


Figure 1: Network data flow for traditional (a) and multi-agent (b) load balancing systems.

A *traditional load balancing system* consists of a single load balancer and a set of servers as presented in Figure 1, a). Accordingly, the input flow is balanced among servers and resulting output traffic is redirected directly to users. Beside obvious performance benefits, a design like this also increases fault tolerance. The failed server can easily be replaced by redirecting its traffic destined among other active servers. Server and service failures are detected by periodic server/service checking. *Multi-agent load balancing systems*, shown in Figure 1, b), enhance scalability by employing an arbitrary number of load balancers. They require a preceding load balancer which distributes input traffic between distributed LB systems. Load balancing at this level can employ coarser distribution policy without any impact on underlying systems. Most often round-robin DNS solution is used to assure geographical distribution of traffic. The main benefit is the possibility to change the size of LB cluster. The LB software can be positioned at any location. Load balancers and servers can be distributed across the Internet, most often geographically. This design makes fast fault detection and recovery possible by enabling LB agents to constantly monitor system activities. The main benefit compared to the previous approach is the fact that servers behind the failed load balancer can be still used by other balancers. A multi-agent approach may utilize an arbitrary number of load balancers and servers.

3 Fault-Tolerance in Agent Societies

In theory, fault tolerant systems (chapter 7. in [7]) should not have a single point of failure and should be resistant to any failure, including the hardware and software ones. In reality we aim to achieve sufficiently high degree of fault tolerance. According to [10], fault tolerant services must be designed to achieve high availability, safety, maintainability, and reliability. Availability is defined by the percentage of time during which a system is operating correctly and is available to perform its functions. Safety refers to the situation where a system temporarily fails to operate correctly but nothing catastrophic happens. Although web services do not fall into the same safety category as, for example, nuclear power plants, one can easily understand the importance of safety considerations when designing web-based services. Maintainability refers to how easily a failed system can be repaired. A highly maintainable system may also show a high degree of availability, especially if failures can be detected and repaired automatically. Finally, reliability refers to the property that a system can run continuously without failure. If a system goes down for one millisecond every hour, it has an availability of over 99.9999 percent, but it is still highly unreliable. Similarly a system that never crashes but is shut down for a week every year has high reliability but only 98 percent availability.
When developing MASs developers often overlook the importance of fault-tolerant software development. Sometimes, especially in *open systems*, it is impossible to enforce fault-tolerant developing strategies since we do not control the development process. Designers can only enforce fault-tolerant strategies through selected multi-agent architecture design. *Complex systems* are by definition difficult to comprehend and are often affected by hard-to-spot faults. Sheer size of such systems prevents complete and systematic testing of system functionality. System-wide fault-tolerant models are

therefore welcome as additional mechanisms, which increase fault resistance of a MAS. Although one can argue that faults in *ubiquitous systems* are least harmful, we state that each fault can affect the very essence of such systems.

3.1 Fault-Tolerance in Load Balancing Systems

We compare LB systems presented in Figure 1 in terms of reliability. If p is the computer error probability over a given time, M is a number of load balancers, and N a number of servers within the system, then we can estimate overall error probabilities. If we assume instantaneous error detection and response, then error probability is as follows:

- Traditional load balancing system with N servers:

$$p + (1 - p) \cdot p^N$$

- Multi-agent load balancing system with N servers and M load-balancers:

$$p^M + p^N - p^{N+M}$$

Results in Figure 2 show that the traditional load balancing system is substantially less reliable than the multi-agent versions for M=2 and N=10. For small error probability p and large numbers of N and M, multi-agent systems become highly reliable.

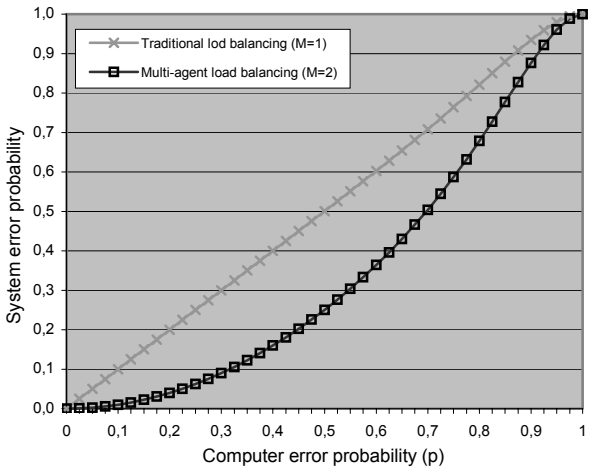


Figure 2: System error probability for M=2 and N=10.

The assumption in previous analysis is that a failed computer is never repaired. This situation is reasonable when we analyze error probability over short intervals of time, e.g. one hour. For longer time periods computers are repaired. If p is the probability of an error over time t and r is average repair time, then the probability that a computer is not operational is $p \cdot r/t$. Most common situation in practice is that either the system is fully operational or one computer is not working. In the latter case, we assumed that all computers have the same failure probability and that the system performance is proportional to the number of working servers. Now we can analyze the ratio of average performance between a system with one failed computer and a system with all working computers for various approaches:

- Traditional load balancing system with N servers:

$$\frac{N\,(N-1)}{(N+1)\,N}$$

- Multi-agent load balancing system:

$$\frac{M}{(N+M)} + \frac{N\,(N-1)}{(N+M)\,N}$$

Multi-agent LB systems shown in Figure 1 b), perform substantially better with one computer down than the traditional version as shown in

Figure 3.

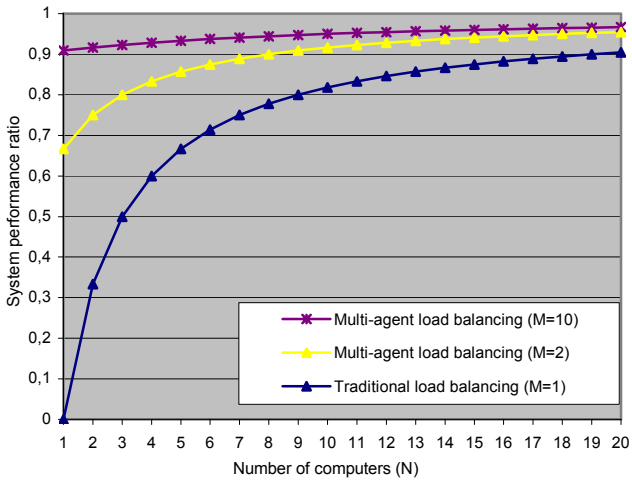


Figure 3: Ratio of average system performance with one failed computer compared to system with all working computers.

Multi-agent systems are prone to failures as any distributed system. Agents and resources may become unavailable due to machine crashes, communication breakdowns, process failures and numerous other hardware and software failures. Most of the work done in fault handling for multi-agent systems deals with communication failures, while the detection and recovery from faults typically rely on the traditional techniques for failure recovery. However, the traditional fault-tolerance techniques are designed for specific situations and the introduction of MAS requires special infrastructural support, such as support for continuous and fault-tolerant agent communication.

3.2 Fault Detection Models

Software faults can be avoided by appropriate software development process. But often, an open agent society does not allow us to enforce software merits on participating agents. In addition, software and hardware

faults are invariant property of all systems, both software and hardware ones. In order to increase fault-tolerance of an agent system, we must introduce a system-wide strategy to detect faults and to take appropriate steps to reduce harmful consequences. The first step therefore is to detect software faults. Most often we want to keep the fault-detection time below some reasonable time limit. To enforce such limit, faults can be detected in a timely manner by periodical checking. The proper check frequency ensures fault-detection within reasonable limits. As faults can severely affect agent functionality, we cannot assume that agents will detect their own faults. They must be detected by some other entities, which raises another interesting problem. How can this entity – call it a check agent – detect faulty operation of other agents? We present some criteria to decide what kind of check models a designer can utilize. The first one is based on check semantics. Accordingly, a check can belong to different semantic levels:

- *Keep-alive check*: a basic query which inspects checked entity whether it is alive or not. Most often this check can be performed without agent knowledge (e.g. system call for process status).
- *Semantic-free check*: a query asks agents to report its status. The task of performing check is delegated to agent where response is only "good" or "bad". Lack of response also indicates agent's fault.
- *Semantic-full check*: a series of queries which test agent functionality. The queries are actually requests for agent functionality. Agent response is parsed and analyzed for proper operation. Check agent must therefore be able to understand the semantics of response.

As each higher semantic level supersedes the lower one in terms of check efficiency, it is often not possible to implement the full semantic check. The functionality of a checked agent can be unknown or simply too complex to implement. It is therefore a designer's obligation to devise appropriate level of check semantics.

Another criterion deals with entity which performs a check. Theoretically, it can be the agent itself. Although possible, such check does not work for a majority of faults as they can affect the operation of an agent. As the sole other possibility, different agent instance must perform a check. However, we can distinguish between different check models, as shown in Figure 4:

- *Buddy*: each agent is checked by it's own check entity.
- *Star*: a group of agents share the same check entity.
- *Parallel*: each agent in a group checks all other ones.

Table 1: Various statistics for different check models.

Check model	Buddy	Star	Parallel
Check channels	n	n	n(n-1)/2
Additional check entities	n	1	0
All entities	2n	n+1	n
Different check entities	n	1	1

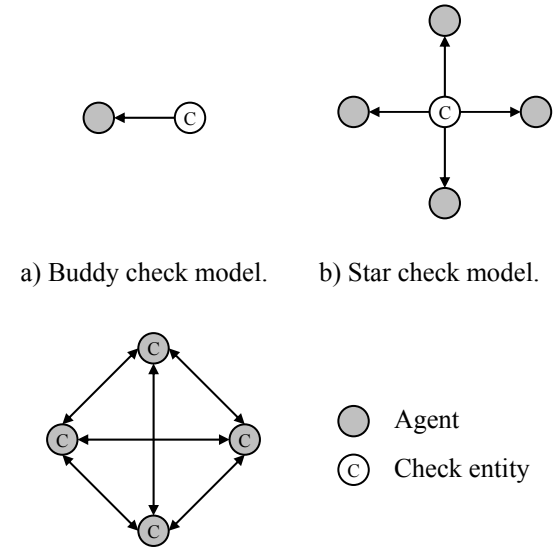


Figure 4: Different check models in multi-agent systems.

Buddy model is the most appropriate when checking a single agent instance with unique functionality within MAS. The downside of this model is a big number of checking entities which is the same as number of agents. *Star model* is efficient where we have a group of agent instances with the same functionality. The same functionality is checked for all agent instances. Obvious advantages are single check entity and efficient resource allocation. The downside is a possible failure of a check entity which suppresses all further check activities. P*arallel model* solves this problem by assigning task of checking to agents themselves; if one agent fails, all others will still perform checking. As agents check each other, they do not need additional check entity. Accordingly their functionality is extended with the check process. This fact also eases the problem of semantic-full check as developers already know the exact functionality of the agent and can therefore design efficient check queries and response analyzers. As a negative side of this model one must mention increased communication-related resources as agents connect with each other. Communication channels must employ efficient delivery methods, e.g.: broadcast on local networks and multicast on wide area networks. Analysis of each model is presented in Table 1, where n represents the number of agents within MAS.

3.3 Related Work

A large number of techniques for fault-tolerance can be found in the traditional database and distributed systems literature. Most of these recovery methods [1] primarily focus on replication techniques that permit critical system data and services to be duplicated as a way to

increase reliability. However, this paper focuses on agent-oriented approaches. Jennings [4] showed that as the world becomes more complex and variable and plans tend to fail more often, agent teams as a whole waste fewer resources and are more robust than self-interested agents. This approach is similar to ours in that both approaches are based on the theory of teamwork. However, we explicitly address the problem of fault-tolerance whereas Jenning's work is more focused towards cooperative problem solving. Kumar and Cohen [8] argued that teamwork might be used to create a robust brokered agent architecture that can recover a MAS from broker failures without incurring undue overheads. Their brokered architecture also guaranteed a specified number of brokers in a large MAS, where agent autonomy can be used to guarantee acceptable levels of quality of service by an agent. Hägg [3] uses external sentinel agents which listen to all broadcast communication, interact with other agents and use timers to detect agent crashes and communication link failures. The sentinels in Hägg's approach analyze the entire communication going on in the MAS to detect state inconsistencies. However, this approach is not realistic for systems with high volume and message frequency. Klein [6] proposes to use exception-handling service to monitor the overall progress of a MAS. Agents register a model of their normative behavior with the exceptional-handling service that generates sentinels to guard the possible error modes. Such exception handling service is also a centralized approach, which is not suitable for scalable distributed systems.

4 A Fault-Tolerant Multi-Agent System

In this section we briefly present a fault-tolerant multi-agent LB system. Its architecture is presented in Figure 1 b). We designed 12 different agent classes. Depending on the agent class, an agent instance can reside on a server, load balancer, management computer, or on each computer within a cluster. A schematic diagram for agent connections is presented in Figure 5. Server-based agents are responsible for handling service and server-related activities: full service/server administration, gathering of service/server-related statistics, and synchronization of service data. Agents hosted on LB computers are LB-oriented and cluster-oriented ones. They perform important activities, such as: checking activities regarding services, servers and load-balancers together with the control of LB software, and enforcement of desired LB policy. Agents with management status handle various cluster-related activities. They provide global configuration together with user interface, report errors, and perform monitoring of agents. Table 2 presents all agent classes with their names, optional acronyms in parenthesis and locations - all in the first line and short descriptions afterwards.

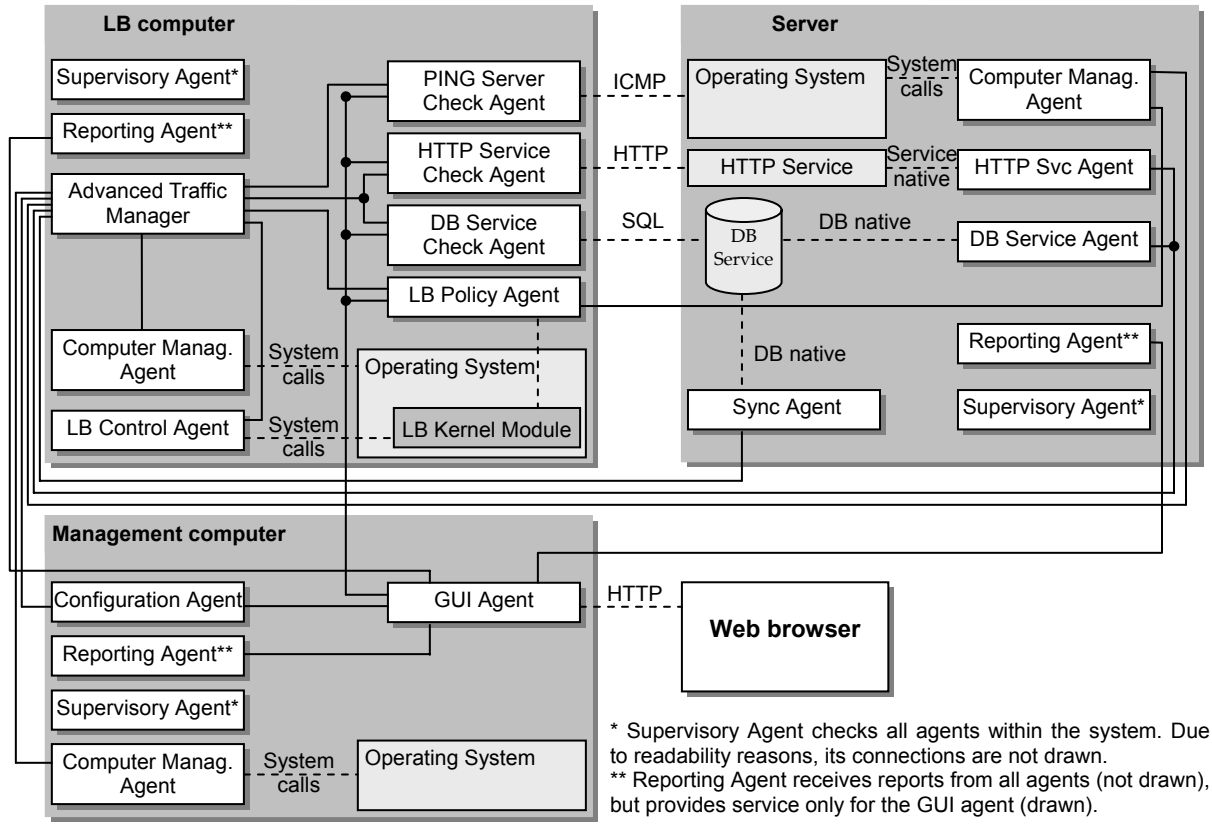


Figure 5: Agent interdependencies and structure of multi-agent LB system.

Table 2: Descriptions of agents in a multi-agent load balancing system.

Advanced Traffic Manager (ATM), LB based. A central cluster management agent. It handles the whole cluster organization and actively checks other ATMs for proper operation. It controls LB software according to configuration, assigns servers and services to LB software, and periodically checks their operation. In case of failure, the server or service is immediately marked as inactive.
Service Management Agent (SMA), server based. A service-specific agent, which performs all service-dependent administration. Its primary task is to setup, start, and stop specific services. It can also retrieve service specific-metrics (e.g. statistics of http accesses).
Synchronization Agent, server based. Takes care of synchronization of service-based data. Its implementation is service specific.
LB Policy Agent, LB based. Assigns weights in LB software according to the desired LB policy. It must monitor server-based metrics (such as CPU load and number of network connections on servers) and compute weights to distribute traffic appropriately.
LB Control Agent (LBA), LB based. Acts as a translator to LB software translating agent commands to specific commands of LB software and reporting status changes of the LB software.
Computer Management Agent, cluster based. Controls various computer-related tasks. It can setup network interfaces and report various metrics (such as CPU load, number of network connections, memory consumption and amount of free disk space).
Server Check Agent, LB based. Checks if a server is working properly. It is possible to implement several different server checks.
Service Check Agent, LB based. Checks if a service is working properly. Each service demands a different instance of service check agent.
Configuration Agent, management based. Keeps configuration and controls simultaneous access to it. It also broadcasts all recent configuration changes to the agent system.
GUI Agent, management based. Acts like a http server for users, and an intermediate agent to the agent system. Its task is to provide web-based user interface for cluster management including access to configuration, messages, errors, and up-to-date information about servers, services, and LB software.
Supervisory Agent, cluster based. Starts, checks and terminates agents within one computer. According to role defined in configuration it must start or stop agent operation of each computer within cluster. In order to deal with unexpected software errors, a special agent was assigned to periodically check the health of running agents. In case of no response, the failed agent is forcefully terminated and restarted.
Reporting Agent (RA), cluster based. Reports various messages and errors to user (via GUI agent).

4.1 Fault-Tolerant Design Issues

Since the fault-tolerant computing paradigm expects failures as a rule and not as an exception, the system must be able to reasonably cope with agent failures. To systematize fault-related activities we introduced different importance levels together with its fault-tolerant design principles. Table 3 presents all importance levels together with corresponding agent classes. Consequently, our design considers the following four levels of agent importance:

- **Core agents** perform tasks essential for proper functioning of the system which stops or is operating erroneous without them. An example for that would be the Advanced Traffic Manager that controls vital cluster management activities.
- **Support agents** carry out tasks regarding management of services, servers and agents. They are needed only for startup and shutdown activities or for reconfiguration of a cluster. For example, the lack of Service Management Agent would prevent the cluster to start or stop certain service but the system would remain stable.
- **Regulative agents** perform partial cluster optimization, and are not critically required to run the system. For example, all checking agents are optional; the system is working also without them. However, the system is thus unable to detect failures of servers or services.
- **User agents** are needed only for users to access cluster management. Its inexistence does not impact the operation of the system.

Table 3: Agent classes with corresponding importance levels. Different levels of shading illustrate importance levels.

Core agents (Critical importance)	**Support agents** (High importance)	**Regulative agents** (Medium importance)	**User agents** (Low importance)
Advanced Traffic Manager	Computer Management Agent	LB Policy Agent	GUI Agent
LB Control Agent	Service Management Agent	Server Check Agent	Reporting Agent
Configuration Agent	Supervisory Agent	Service Check Agent	
	Synchronization Agent		

Table 4: Actions following possible problems

Entity	Possible problems	Action	Semantic level	Check model
agent	agent crash agent malfunction	restart agent terminate & restart agent	Semantic-less	Star
service*	service not accessible service crash service malfunction	try to setup appropriate network interface restart service exclude it from operation	Semantic-full	Star
ATM	computer crash LB software malfunction	exclude it from operation exclude it from operation	Keep-alive	Parallel
server**	computer crash server malfunction	exclude it from cluster force server shutdown and exclude it from cluster	Keep-alive and Semantic-full	Star

* Each reported problem forces service on a computer to be excluded from cluster. It is included later if service checks report that action was successful.

** Each server with reported problems is excluded from cluster. It is included later if server checks report that action was successful.

To increase the fault-tolerance of our system, we incorporated failure prevention for different entities within MAS. The anticipated actions on possible problems together with their semantic level and check model are summarized in Table 4.

Instances of Supervisory Agent class perform basic agent monitoring. Its main function is to monitor agents for proper operation. Since implementing semantic-full check for arbitrary agent is too time consuming and agents can spawn several processes thus forbidding keep-alive queries with system process routines we implemented semantic-less checking. On each check query, agents respond with their state. An agent that does not respond to a check query is considered failed. Because of relatively high number of running agent instances, we designed a distributed version of star check model. With it one instance Supervisory Agent reside on each computer within MAS and monitors only local agents; i.e. it checks all agent instances residing on the same computer. A failure of a single Supervisory Agents therefore only affects monitoring activities on one computer. Service, server, and ATM failure detection is also achieved by periodical checking of each entity. Since the number of services and servers is arbitrary high we chose a check model with the lowest communication-related overhead: a star check model. All checking is performed on an active load-balancer by Service/Server Check Agents, as shown in Figure 5. Servers can be checked with ICMP keep-alive queries (i.e. ping) or semantic-full queries utilizing Computer Management Agent hosted on servers. All services are checked with semantic-full queries (e.g. we implemented HTML queries for checking web services). The most important agent within our system, ATM, was designed with a special protocol to enable mutual checking of ATMs. According to parallel model, each ATM checks other active ATMs. Active ATMs announce their active presence by periodically broadcasting "heartbeat" messages over LAN that also replaced $O(n^2)$ communication channel utilization with only O(1). If, for some known period of time, this message is not received by other ATMs, the sender is considered failed. In this case, elections start and the newly elected ATM begin acting as a primary, while previous active ATM is demoted. Our design allows arbitrary number of ATMs to act as primary while other act as backup ones. Each ATM owns its unique identification number (ID), while each possible active position is represented as one slot and backup ATMs take part in elections for each slot separately. Active ATM for one slot cannot participate in elections for another slot. The voting protocol for each slot is described in Figure 6.

When ATM is elected as active, it starts advertising virtual IP of a running cluster. In LAN environments this can be achieved by advocating virtual IP number with technique called ARP spoofing, which involves constructing forged ARP replies. With this technique we can convince the network gateway that IP of nonexistent computer is actually owned by an existent computer – a load balancer in our case. This enables us to have a virtual IP number, which is associated with an active LB. The traffic destined to virtual IP number is redirected to active ATM which hosts operating LB software.

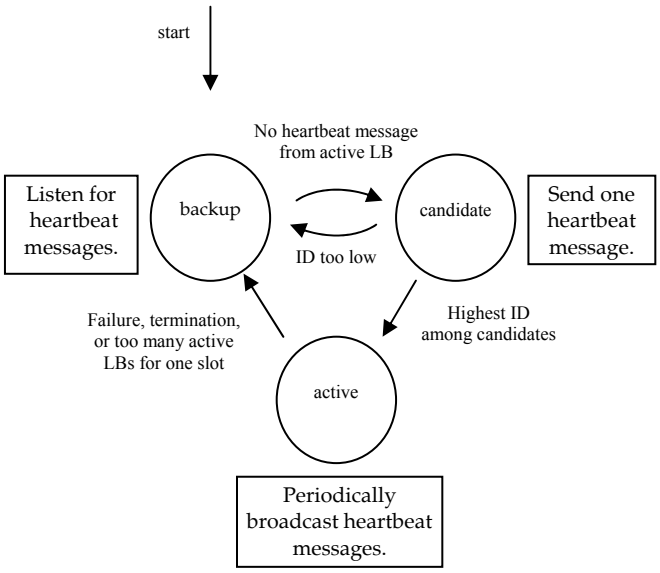


Figure 6: State diagram for voting protocol used in elections for the active load balancer.

The design of Configuration Agent proved to be quite demanding. To allow redundancy, Configuration Agents must be able to operate simultaneously, thus the task of designing a Configuration Agent is similar to designing a distributed database. Having known implications of such decision, we softened parallelism requirements. The final design allowed only one active Configuration Agent, while others are running in a backup mode. To achieve configuration consistency, all configuration changes can only be committed to the active instance. All backup instances synchronize its configuration with the active one. In case that the active agent fails, configuration can still be retrieved from the backup ones. The location of the active instance is left to the system administrator so as to allow full control over the current configuration computer.

The separated design of agent classes and further separated implementation of agent instances introduces one important improvement over traditional programming. With the latter, developers must corporately develop a big and complex program. Although its design can be modular and developers develop distinct modules, it poses several implementation-related problems. For example, as developers introduce errors, the development process is stopped for all developers working on the same program. Sometimes, the big and complex structure prohibits deep understanding causing developers to overlook hidden module dependencies and introduce hard-to-detect bugs. In the development of MAS developers are more evenly distributed on development of separate agent instances, which also tend to be smaller implementation tasks. Typically one developer develops the whole agent, thus removing inefficiencies presented with the traditional programming. Consequently, the development process is more straightforward, thus faster and less error-prone thus also more fault-tolerant.

5 Conclusion

We have theoretically analyzed fault-tolerance for traditional and multi-agent load balancing systems. Our analysis show that multi-agent load balancing systems formally introduce important improvements such as lower system error probability and better average performance in case one computer is not working. We also presented different semantic check levels and check models. Our analysis reports applicability conditions for different semantic levels and models. The presented ideas are then described in an application of multi-agent load balancing system. We show that presented models do not increase fault-tolerance of single agent instance but clearly improve fault-tolerance of a whole MAS.

It is important to be aware of the advantages and disadvantages of agent and non-agent approaches, but the most important is whether advantages prevail. For load balancing, our theoretical analysis and practical experiences both indicate that advantages of MAS LB systems evidently overweight observed disadvantages.

References

[1] K. P. Birman, editor, "Building Secure and Reliable Network Applications," Part III, Reliable Distributed Computing, chapters 12-26, 1996.

[2] E. H. Durfee, V. R.Lesser and D. D. Corkill, "Trends in Cooperative Distributed Problem Solving," in IEEE Transactions on Knowledge and Data Engineering, KDE-1(1), pp. 63-83, 1989.

[3] S. Hägg, "A Sentinel Approach to Fault Handling in Multi-Agent Systems," In Proceedings of the 2nd Australian Workshop on Distributed AI, Cairns, Australia, 1997.

[4] N. R. Jennings, "Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems using Joint Intentions," Artificial Intelligence. 75(2), pages 195-240, 1995.

[5] N. R. Jennings, "On agent-based software engineering," Artificial Intelligence 117, pp. 277-296, Elsevier, 2000.

[6] M. Klein and C. Dellarocas, "Exception Handling in Agent Systems," Autonomous Agents '99, Seattle, 1999.

[7] S. Tanenbaum, and M. van Steen, "Distributed Systems: Principles and Paradigms," Prentice-Hall, 2002.

[8] S. Kumar and P. R. Cohen, "Towards a Fault-Tolerant Multi-Agent System Architecture,", in Proceedings of The Fourth International Conference on Autonomous Agents, 2000.

[9] A. Pnueli, "Specification and Development of Reactive Systems," in Information Processing 86, Elsevier/North Holland, 1986.

[10] P. Verissimo and H. Kopetz, "Design of distributed real-time systems," in Shape Mullender, editor, Distributed Systems, chapter 19. Addison-Wesley, 2nd edition, 1995.

Empirical Assessment of Methods for Software Size Estimation

Aleš Živkovič, Marjan Heričko and Tomaž Kralj
University of Maribor, Faculty of Electrical Engineering and Computer Science, Institute of Informatics
Smetanova 17, SI-2000 Maribor
ales.zivkovic@uni-mb.si, http://lisa.uni-mb.si

Keywords: Software metrics, Function Points, Software Size Estimation, Empirical Analysis

Received: July 20, 2003

In the software industry, many projects fail due to both the misjudgment of a project's size and faulty estimates correlated to this elementary metric. Several methods for software size estimation are present. The Function Points Analysis (FPA) method, however, is most frequently put into practice. After Albrecht introduced the FPA method, several variations evolved. All methods share the same fundamental idea, but differ in procedural steps and metric units. A descriptive approach is usually used for method comparison. To avoid the weaknesses of a descriptive approach, a mathematical model is defined and used for theoretical comparison. The complexity of the mapping functions prevent detailed comparisons -- consequently only general characteristics become evident. Characteristics exposed with a formalization of the rules were further studied in different test scenarios using historical data from past projects. Empirical results showed some limitations of the mapping function and anomalies in the data set used. The possible reasons for deviations in the data set were also analyzed.

1 Introduction

Software size estimation is a crucial element in a project manager's decision-making process, with regard to the project's duration, budget and resources. In the past, different methods were developed. Albrecht introduced the function point analysis method in 1979 [1], since then it has been the target of many scientific studies [4, 5, 6]. Some modifications have also been made resulting in new methods like Feature points, Full Function Points, Function Weight, Function Bang, Mk II Function Points Analysis, COSMIC-FFP and NESMA.

A comparison of different methods, based on verbal descriptions, lack the formalism needed to understand and compare them. In this paper, a mathematical foundation for describing the methods is established first, and then three popular methods are mapped into the universal form and compared. To compare the mapping functions, the empirical method is used. The paper is divided into four sections. In the first section, the methods for software size estimation are briefly presented. The subsequent section introduces the formal model for representing software-sizing methods. The third section describes test scenarios and presents results. The conclusion and plans for future work can be found in the last section.

1.1 Function Points

The idea behind function points is quite simple [2]. Every information system processes some data that can be stored in the application database or is gained from external applications. Four operations are performed on data records: create, read, update and delete. Besides that, information systems use several query functions for data retrieval and report construction. Each record consists of several fields of basic data types or another record that can be further decomposed. The FPA method quantifies: the number of fields in each record, the distinct operations performed on these records, and the number of these operations that are necessary to perform a business function. The sum over all business functions, multiplied with some empirically determined weights, represents unadjusted function points. The final calculation is made using a value adjustment factor (VAF) that measures system complexity. Figure 1 shows an overview of the tasks performed within the scope of the FPA method.

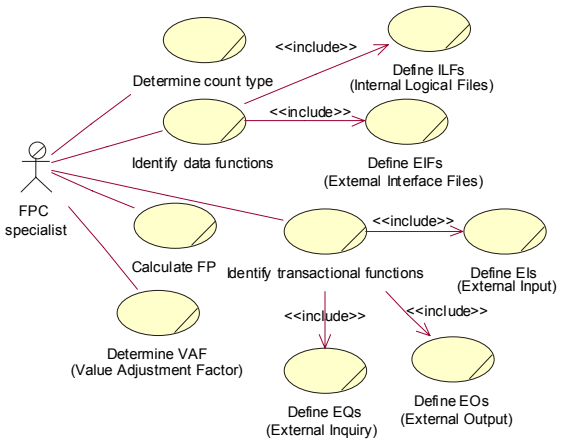


Figure 1: Business Use Case diagram for FPA method

1.2 COSMIC-FFP

The COSMIC-FFP method [10] reached standardization in 2003 as ISO/IEC 19761 and is the only method in accordance with ISO/IEC TR 14143 [7]. Its approach to size measurement is different from the original FPA, since data elements do not contribute directly to the size. The focus of interest is on data movement, which is defined by units of measure called Cosmic functional size units (Cfsu). In [10] the conversion factor for function points is given based on a sample application portfolio of 14 applications from two different systems. In general, the conversion factor is close to one, when comparing unadjusted function points. Methods distinguish between four different data movements (entry, exit, read and write), and the sum of their size represents the size of the system measured. Beside raw measurement rules, the method clearly defines its applicability in different circumstances (e.g. software domain, project phase). Its popularity among practitioners is growing.

1.3 Mark II FPA

In 1988, Charles Symons developed a variation of the FPA method [3] adding several new steps into the measurement process. Additional steps are bound into calculating the effort, productivity and influence in the technical complexity of a specific solution. A functional size itself is calculated as the weighted sum over all logical transactions of the input data element types (N_i), data entity types referenced (N_e) and output data element types (N_o). For the weights, the industry average is used with values Wi=0.58, W_e=1.66 and W_o=0.26. Compared to the original FPA method, the major difference is that MK II FPA is a continuous measure with linear characteristics. Therefore MK II FPA produces increasingly higher size estimates for projects with more than 400 function points. The primary domain for MK II FPA is business information systems. If applied to other domains, special attention has to be given to components with complex algorithms, since sizing rules do not take into account their contribution. For use with real-time systems, additional guidelines may be necessary [3].

2 FPC formal model

All methods for software size estimation lack formal foundations in their origin descriptions. There were some attempts [8, 9] in the past to add formalism to functional size measurement. Fetcke's model is applicable to different methods since it introduces an additional level of abstraction. The approach proposed by Diab et al. is designated to COSMIC-FPP and has a specific purpose. In our research, the model defined by Fetcke is used as a basis and further refined by the definition of a mapping function.

2.1 Generalized representation

According to measurement theory, every measurement can be represented as a function that maps empirical objects into numerical. The FPA method defines a function that maps a software system into a number. That number represents the size of the system. Since the FPA method is technologically independent, it introduces its own concept for representing a software system. The abstraction of a software system is data oriented and has two steps.

1. The software documentation is transformed into elements defined by the method.
2. Method elements are mapped into a numerical value representing the size of the system expressed in function points.

The procedure is presented in Figure 2 as a UML activity diagram. It shows a specific example of where Software Requirements Specifications (SRS) serve as an input to the FPA elements identification process. Elements are identified according to rules, and the outcome is a data-oriented abstraction of the software system. The second activity represents the mapping function. Several tables are used for the transformation of a separate element count into function points. The final result is the number of function points.

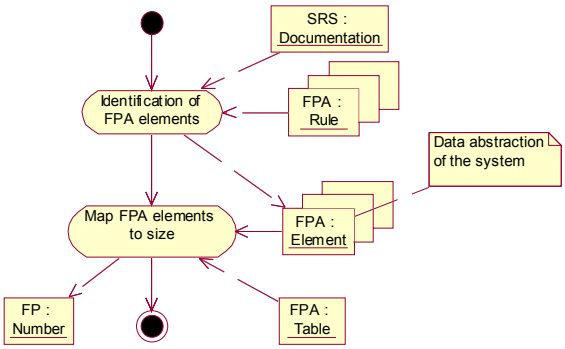


Figure 2: Data abstraction steps for FPA method

2.2 Data-oriented abstraction

Different methods enumerated in the introduction use different names for data abstraction elements; the rules for element identification are different, mapping functions also differ. However, similarities exist that can be described by the following core concepts:

- The user concept covers the interaction between a user and the system.
- The application concept represents the whole system as an object of the measurement.
- The transaction concept is the logical representation of the system's functionality. Transaction is the smallest independent unit of interest.
- The data concept deals with the subject of change within the system. The data element is the smallest unit of user observation.
- The type concept simplifies data handling via the abstraction of individual data elements.

On a higher level of abstraction, an application is represented with data and transactional types. The data

type is a set of data elements handled within the system. The transactional type is a sequence of logical activities. Fetcke defined seven classes of logical activities [8]:

- *Entry activity*. The user enters data into the application.
- *Exit activity*. Data is outputted to the user.
- *Control activity*. The user enters control information data.
- *Confirm activity*. Confirmation data is outputted to the user.
- *Read activity*. Data is read from a stored data group type.
- *Write activity*. Data is written to a stored data group type.
- *Calculate activity*. New data is calculated from existing data.

In Figure 3, a UML class diagram for data-oriented abstraction can be found. Based on the abstract presentation, mapping for a specific method can be made.

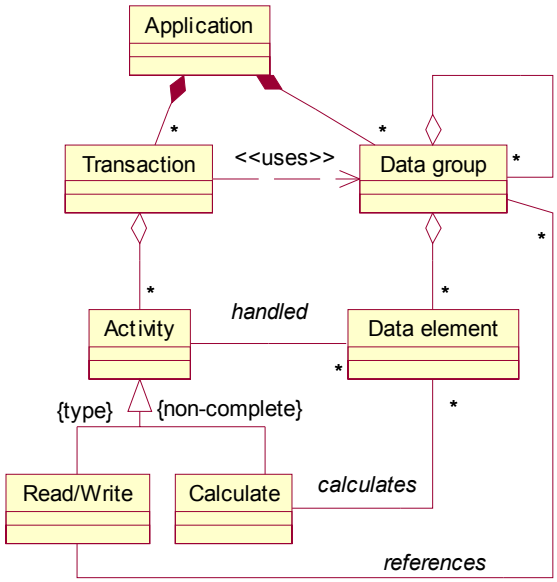


Figure 3: Relationship between FPC elements

2.3 Generalized structure

In this section we summarize the formal representation of concepts described in the previous section. An application closure H is defined as the vector of τ transactional types T and σ data group types F.

$$H = (T_1, \ldots, T_\tau, F_1, \ldots, F_\sigma) \qquad \textbf{(E 1)}$$

The transactional type T_i is a vector of activities

$$T_i = (P_{i1}, \ldots, P_{in_i}) \qquad \textbf{(E 2)}$$

An activity is further described by four attributes:

- its class $\theta_{ik} \in$ {Entry, Exit, Control, Confirm, Read, Write, Calculate},
- for read and write activities, the data group type referenced $r_{ik,}$
- the set of data elements D_{ik} handled and
- for calculate activities, the set of data elements calculated C_{ik}.

In the equation E3, i can have values from 1 to τ and represents the transaction activity it conforms to. k runs from 1 to n identifying activity within the transaction.

$$P_{ik} = (\Theta_{ik}, r_{ik}, D_{ik}, C_{ik}) \qquad \textbf{(E 3)}$$

The data group type F_j is a set

$$F_j = \{(d_{j1}, g_{j1}), \ldots, (d_{jr_j}, g_{jr_j})\} \qquad \textbf{(E 4)}$$

where the d_{jk} are data elements and the g_{jk} the designate sub-groups. j can have values from 1 to σ and represents the number of data types. k distinguishes between data elements and can have values from 1 to r.

2.3.1 Representation of the mapping function

In the previous section, a formal representation of transactional and data types was introduced. The system is composed of different data and transactional types. The number of data and transactional types, and their attributes, contribute to the size of the software system. Some methods also define the third component that has an influence on software size, and the technical complexity of the solution. The universal function that maps application attributes into size is therefore:

$$FPC(a) = \left(\sum_i FPC_1(t_i) + \sum_j FPC_2(f_j) \right) * FPC_3(TC) \qquad \textbf{(E 5)}$$

where

FPC(a) is the function that maps attributes of the application a into software size.
$FPC_1(t_i)$ is the function that maps transactional type t_i into size.
$FPC_2(f_j)$ is the function that maps data type f_j into size.
$FPC_3(TC)$ is the function that maps technical complexity of the anticipated solution for application a into a factor.

The total value for an application size is the sum of both parts multiplied by the factor of the solution's complexity. The factor can reduce or increase the overall size. However, it is not clear if the factor actually measures raw application size or is an attribute of the implementation and should be part of the function that

maps size to effort. In this research, the function of FPC_3 is not examined.

A generalized structure can now be used to define different methods. First, we will use it for representing the original FPA method.

2.3.2 Mapping for the FPA method

Since data functions from the FPA method correspond to data element type (F), data element type (DET) corresponds to data element and record element type (RET) is equivalent to sub-group defined in the generalized structure. The FPA method distinguishes between internal and external data requirements; generalized representation, however, defines more activity types than the FPA method. Therefore, we define external interface files (EIFs) as a data type that cannot be used in write type activities.
The mapping of the transactions is a bit more complicated and is summarized in Table 1. In the table, activities that are allowed in the transaction type are marked with X.

Table 1: Mapping of transactions into activities

		GENERAL STRUCTURE ACTIVITIES						
		Entry	Exit	Write	Read	Confirm	Control	Calculate
FPA	EI	X			X	X	X	X
	EO		X		X			X
	EQ	X	X		X	X	X	

The FPC functions for the FPA method would look like this:

$$FPC_1 = \sum_i W_{EI}(N_d, N_r) + \sum_j W_{EO}(N_d, N_r) + \sum_k W_{EQ}(N_d, N_r)$$
$$FPC_2 = \sum_l W_{ILF}(N_d, N_g) + \sum_m W_{EIF}(N_d, N_g)$$

(E 6)

where the W_{EI}, W_{EO}, W_{EQ}, W_{ILF}, W_{EIF} are functions that prescribe the number of function points for every FPA function identified in the measurement process. Function W has two parameters. For transactional functions, parameters are the number of data element types (N_d) and number of file types referenced (N_r), for data functions parameter N_g is used instead of N_r, representing the number of record element types. Functions W_x are the step functions represented by discrete values with the following range:

W_{ILF}={7, 10, 15}
W_{EIF}={5, 7, 10}
W_{EI}=W_{EQ}={3, 4, 6}
W_{EO}={4, 5, 7}

Given as an example, the step function W_{ILF} is defined as:

$$W_{ILF}(N_d, N_g) = \begin{cases} 7; & ((1 \le N_d \le 19) \wedge (1 \le N_g \le 5)) \vee \\ & ((20 \le N_d \le 50) \wedge (N_g = 1)) \\ 10; & ((1 \le N_d \le 19) \wedge (6 \le N_g)) \vee \\ & ((20 \le N_d \le 50) \wedge (2 \le N_g \le 5)) \vee \\ & ((51 \le N_d) \wedge (N_g = 1)) \\ 15; & ((20 \le N_d \le 50) \wedge (6 \le N_g)) \vee \\ & ((51 \le N_d) \wedge (2 \le N_g)) \end{cases}$$

(E 7)

2.3.3 Mapping for the MKII FPA

In the Mark II FPA method, data groups are called entity types and do not directly contribute to functional size. Therefore, FPC_2=0 in all cases. Logical transactions are broken down into activities. There are only three types of activities in MK II FPA, namely input, processing and output. Table 2 shows mapping for activities defined in generalized form. Notice that MK II FPA does not have an equivalent to the calculate activity, which is due to processing activity deals with existing entities, and which conforms with the read and write activity types.

Table 2: Mapping for MK II FPA

		GENERAL STRUCTURE ACTIVITIES						
		Entry	Exit	Write	Read	Confirm	Control	Calculate
MK II FPA	Input	X					X	
	Output		X			X		
	Processing			X	X			

$$FPC(a) = \sum_j (W_i * N_{di}) + (W_e * N_F) + (W_o * N_{do})$$

(E 8)

where the W_i is the weight for input elements and has a constant value of 0.58, W_o, is the weight for output elements with the value 0.26, W_e is the weight for entities referenced in processing with the value 1.66, N_{di} is the number of data elements used in the input activity, N_{do} is the number of data elements used in the output activity and N_F is the number of entities used in processing.

2.3.4 Mapping for COSMIC-FPP

As described in the introduction, the COSMIC-FFP method defines Cfsu as a unit of measure and introduces a different approach to software sizing. The method counts data movements that can be one of four types: entry, exit, read, and write.

$$FPC = \sum_{i=1}^{n} T_i = \sum_{i=1}^{n} \sum_{k=1}^{4} P_{ik}(F) \quad \textbf{(E 9)}$$

Equation 9 shows the mapping function. The sum across all identified transactions (T_i) is made in the first part of the equation. In the second part, transactions are broken down into activities (P_{ik}), where k runs from 1 to 4, since the method has only four types of activities. With the F in brackets, we have revealed that activity depends on data types, since data is the object of movement. Again FPC_2=0 and only FPC_1 contributes to the application size.

It can be seen from the equations E6, E8 and E9 that FPC functions of selected methods are multivariable, thus further research into them is complex. To observe them in specific situations, we have set a few test scenarios described in the text section.

3 Test scenarios

With methods for software size estimation two kinds of errors are likely to occur: a method error and a measurement error. A method precision is not formally defined nor statistically proven. Approximate values can usually be found that imply method accuracy. Since the behavior of the FPC function is dimmed, it is difficult to predict results in all circumstances. To analyze the basic characteristics of the FPC function for three selected methods, we have to construct three diverse scenarios. The findings help us choose the right method for the given problem domain.

A measurement error can be identified via an analysis of historical data. In the second part of our research, only the original FPA method was used to estimate the size of eight applications. The data gathered were used as the small dataset and compared with the industry average. The deviations in the dataset are analyzed in the second part of this section.

3.1 Empirical comparison of different FPC functions

In the first part of our research we set up three different scenarios, applied different methods and compared the results in order to find deviations between methods' FPC functions. In this research, we decided to apply the original FPA method, MK II FPA and COSMIC-FFP. For the first case we chose only one requirement from the bigger payroll application. The purpose of the selected requirement was to print out specific data in order to monitor the final account for a specific period of time. We named this function Account Control. Let's summarize the measurement technique for all three methods. Because we have chosen only part of the whole application, applying COSMIC-FFP, some steps were excluded from the counting procedure. We followed only the necessary steps in performing the task. In applying the original method and MK II FPA, the Value Adjustment Factor (VAF) was not calculated. Therefore, size is expressed in unadjusted function points (UFP). Lokan discouraged the use of VAF, according to his empirical analysis of FP adjustment factors [4]. The VAF was found not to improve the relationship between FPs and effort. For most projects, the VAF does not result in much change to the function point value [11]. To get considerably accurate results with the original FPA method, we added the contribution of data functions to the value of unadjusted FPs. Since only part of the system was sized, equation E13 was used. FPC_F represents the contribution of data functions and is calculated from the total contribution of data functions (FPC_2) divided by the number of data functions (l and m) and multiplied with the number of referenced data types (N_r).

$$FPC_F = \frac{FPC_2}{l+m} * N_r \quad \textbf{(E 10)}$$

Table 3: Results summary for the test scenarios

	SIZE IN FUNCTION POINTS		
	FPA	MK II FPA	COSMIC-FFP*
TS1	5. 5	10.5	5
TS2	14	33.1	18
TS3	3	3.2	5

* 1Cfsu treated as 1 FP [10]

Table 5 summarizes the results for all test scenarios. The row labeled TS1 shows results from the first test scenario, Account Control function. With 10 function points, MK II FPA produced twice the results of the other two methods. The reason for this lies in the three referenced entities that added 5 FP.

In the next scenario we measured function behavior, using many data element types (DET). In the original FPA method, the increased number of data element types did not influence the contribution of the transactional function to the final size. By comparison, MK II FPA reacts to all changes with a greater amount of FPs. In the example, personal data has to be entered for an employee. The number of attributes was set to 51. The second row in table 5 summarizes the results. The FPA method produced the smallest size, since its FPC function is a step function that cannot follow growth in data elements and referenced data types. The COSMIC-FPP method follows the change in the number of data elements with its read activity, however data elements are grouped for the entry activity. Consequently the final results are smaller than with MK II FPA. The MK II FPA has produced the greatest number of function points. The changes in the number of data elements directly influence the final amount with the factor 0.58. In our opinion, however, it is difficult to predict, in cases like this, how much more effort is necessary when we increase the number of data element types.

Our last test deals with real time applications. The original FPA method is already known to produce non-accurate results for applications in this group. How about the other two methods? We measured the size of

applications that regulate the temperature in a building. The results can be found in the third row (Table 5).
We can argue that only the COSMIC-FPP method produced a correct result, since the result of the Mk II FPA is almost the same as the result of the original FPA method, known not to produce accurate results within real-time systems. Therefore, the warning concerning the real-time application domain in the MK II FPA manual has to be taken seriously. The difference would be even greater if more sensors gathering data and controlling output were introduced.

3.2 Influence of subjectivity

In the second part of our research we selected eight different applications developed by the same company. Although it would be better to apply all three methods, only the original FPA method was used. The applications under consideration had to be developed in the same environment, for the same target platform, with the same tools and the same group. Consequently, only eight applications satisfied the criteria. We have marked applications with letters from A to H. Table 3 briefly illustrates all the selected applications.

Table 4: Short description of selected applications

A	Most recently developed application and currently in use. Analysis and design were carried out in a systematic manner. Therefore all the documentation was available.
B	A lot of documentation exists, describing the current state of the application. It was recently enhanced.
C	Older application with incomplete documentation. It was changed many times in the past without making the appropriate corrections in the corresponding documentation.
D, E, F	Applications had some kind of documentation that was not precise enough to perform the count.
G	Newer application with good documentation.
H	The largest application developed as an answer to the Y2K problem. It was developed under stress and lacks proper documentation.

In cases D, E, F and H we have used GUI forms and E-R model to perform the count. For these applications, the counting specialist took additional measures; consequently the counting speed was reduced.

Table 5: Results for the original method

Application	Number of FPs	Effort (hours)	PDR (h/FP)	PDR_{ISBSG}
A	102	726	7.1	3.6
B	141	1000	7.1	4.0
C	128	800	6.2	3.9
D	254	1600	7.5	5.1
E	472	3000	6.3	6.5
F	485	3000	6.2	6.5
G	624	4400	7.1	7.2
H	719	6000	8.3	7.6

Table 2 depicts the counting results for all eight applications. In the second column, the application's size is expressed in FPs showing that A, B and C are smaller applications; applications G and H are larger. In the third column, we can find the number of hours spent on implementation. The last two columns show the value calculated from the number of FPs and hours spent on implementation. It represents the amount of time spent implementing one FP and its so-called Project Delivery Rate (PDR). PDR_{ISBSG} is the value calculated from the ISBSG repository [11] data using equation E10 with the values 0.587 for constant C, and 0.390 for constant E. The FPC is the function point count expressed in function points. The calculated value represents the "normal" PDR for the project of a specific size, according to the repository's average. Comparing these two values, the performance of projects A, B, C and D is quite unsatisfactory, with deviation from 50% to 100% in hours spent implementing one function point.

$$PDR = C * FPC^{E} \quad \textbf{(E 11)}$$

The reason for the deviation in the results could be one of the following:

- An error occurred within the counting procedure, resulting in less function points.
- The project group performance was actually below the industry average.
- The documentation for the application does not include all features developed.

From the theoretical point of view, only the first case is interesting. Let us assume that the types of some elements were mixed. Equations E11 and E12 calculate the error. In the equation E11 the error for data functions is calculated. To get concrete numbers we need a ratio between data element types.

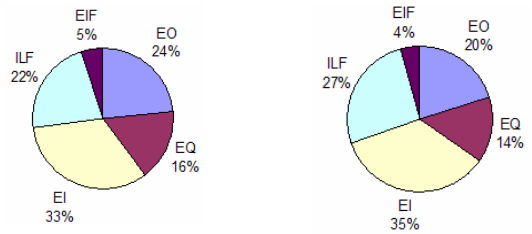


Figure 4: Relationship between FPA elements

The left graph in Figure 4 shows the ratio between FPA elements from the ISBSG repository with 238 projects in the sample, on the right is the graph for our sample. According to the industrial average, there are less then 20% of external interface files in the data functions for standalone applications developed from scratch. If we count all elements as internal logical files (ILF) the error made is around 5 %. In the opposite case, when we neglect ILFs, the error is 35 %.

$$E_{DF} = \frac{\sum_i W_{ILF}(F_i)}{\sum_j W_{ILF}(F_{ILF\,j}) + \sum_k W_{EIF}(F_{EIF\,k})}$$

$$E_{EIF} = \frac{N_{EIF}}{N_{DF}} * \frac{W_{EIF}(F_{ILF})}{W_{ILF}(F_{EIF})} \cong \frac{N_{EIF}}{N_{DF}} * \bar{e}_{EIF} = 0.185 * 0.3 = 0.0555$$

$$E_{ILF} = \frac{N_{ILF}}{N_{DF}} * \frac{W_{ILF}(F_{ILF})}{W_{EIF}(F_{ILF})} \cong \frac{N_{ILF}}{N_{DF}} * \bar{e}_{ILF} = 0.815 * 0.44 = 0.3586$$

(E 12)

The equation E12 calculates the maximal error for transactional functions. Since external inputs and external inquiries have the same weight, they are treated equally. If external inputs and external inquiries are both neglected and all transactional functions are treated as external outputs, errors could be as high as 16%. If external outputs are mixed with external inputs or external inquiries, the error is around 7%. The final numbers are specific for the case presented in the paper and must be recalculated for other types of applications.

$$E_{TF} = \frac{\sum_i W_{EI,EQ}(T_i)}{\sum_j W_{EI}(T_{EI\,j}) + \sum_k W_{EO}(T_{EO\,k}) + \sum_l W_{EQ}(T_{EQ\,l})}$$

$$E_{EI,EQ} = \frac{N_{EI,EQ}}{N_{TF}} * \frac{W_{EI,EQ}(T_{EI}\,/\,T_{EQ})}{W_{EO}(T_{EI}\,/\,T_{EQ})} \cong \frac{N_{EI,EQ}}{N_{TF}} * \bar{e}_{EI,EQ} = 0.67 * 0.25 = 0.1676$$

$$E_{EO} = \frac{N_{EO}}{N_{TF}} * \frac{W_{EO}(T_{EO})}{W_{EI,EQ}(T_{EO})} \cong \frac{N_{EO}}{N_{TF}} * \bar{e}_{EO} = 0.33 * 0.2 = 0.066$$

(E 13)

3.3 Method comparison

All compared methods use the same type of abstraction, based on requirements document for the software system. However, the FPC function that maps identified elements to the size is different and difficult to compare mathematically. Therefore we have set three diverse test scenarios to evaluate a method's performance. To be able to observe a function's behavior, test scenarios were simplified and may be unusual for real-world application. In the first test case, the characteristics of functions were analyzed. In the documentation for MK II FPA and COSMIC-FPP, the function is described as linear with respect to the number of data elements. The original FPA method measures the size of the transactional function according to its complexity. Transactional functions can have 3 to 7 function points regardless of their simplicity or complexity. The graph in Figure 5 shows the behavior of unadjusted function points for external inputs (EI) and external outputs (EO) with respect to the number of data element types for MK II FPA and the original FPA. From the graph it is easy to see when the threshold for the original FPA method is reached and higher values of data element types (DET) do not influence the number of unadjusted function points (UFP). In the COSMIC-FPP method, the behavior of the function depends on the grouping of data elements. For example, all 51 elements can be treated as one Cfsu in the case of entry activity. On the other hand, they influence the final size through the read activity contribution. The method is more complex to use than the other two compared in the test.

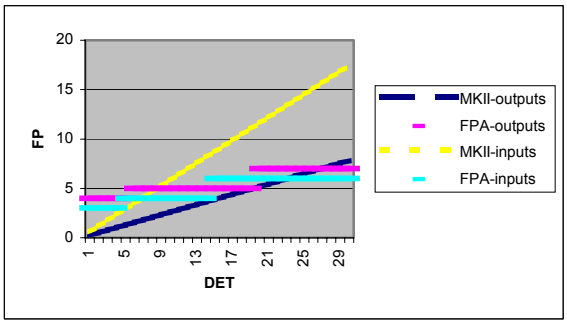


Figure 5: The difference between MK II FPA and original FPA

In the last test of the first scenario, two presumptions were relied upon. The first one was that original FPA does not perform well measuring real-time applications and the second was that COSMIC-FPP does. The results confirmed the generally accepted opinion that MK II FPA and original FPA perform poorly with real-time applications and COSMIC-FPP is the most appropriate method for that domain.

In the second test scenario, an anomaly in the PDR values appeared. We compared original values with values calculated from the ISBSG repository. In the relationships between the FPA elements, it is possible to calculate errors for an element type mismatch. Although the error could rise up to 35 % in our case, the reason for the anomaly was either a poor group performance or incomplete documentation.

4 Conclusion

Software size is an important attribute, which we can use to manage the software development process. It is easy to calculate the effort and costs of a project and to monitor its progress. We can predict the software's size based on experiences from past projects or we can use methods and empirical data. Many projects from the past that relied on a project manager's intuition failed. Thus, we suggest using some method. In this paper, several methods were evaluated and compared in diverse scenarios. The results showed that the consistent use of a selected method in the same environment gives a better understanding of both the project size and the delivery rate. Anomalies in the results become quickly evident and can be analyzed with the help of the general representation of methods for software size estimation, proposed by Fetcke and supplemented with the mapping functions presented in this paper. The mathematical representation of the method and its rules, make them more evident and easier to analyze. The comparison showed that it is important to choose the most appropriate method for the given problem domain to exclude the possibility of anomalies in the results. Our research confirmed that the MK II FPA has some advantages compared to the original FPA method, notably when a lot of DETs are present in the

application. However, both methods performed poorly in the case of real-time applications and system software. The COSMIC-FPP method gives better results with a higher number of FPs.
Function points have suffered a lot of criticism that has discouraged their use in practice. With the data from past projects and with the industry average, both problems could be overcome, namely the problem of early estimates and the human factor problem.
In the future, we will try to apply the second test scenario to another two methods, theoretically compare the mapping functions and express the error.

6. References

[1.] J. Albrecht (1979), *Measuring Application Development Productivity*, Proceedings of Joint SHARE, GUIDE and IBM Application Development Symposium, str. 83-92.
[2.] M. Bradley, ur. (1999), *Function Point Counting Practices Manual, Release 4.1*, International Function Point Users Group (IFPUG), Westerville
[3.] *Mk II Function Point Analysis, Counting Practices Manual*, The UK Software Metrics Association., http://www.uksma.co.uk/public/mkIIr131.pdf
[4.] C.J. Locan (2000), *An empirical analysis of function point adjustment factors*, Information and Software Technology, 42, 649-660
[5.] C. Yau, H. Tsoi (1998), *Modeling the probabilistic behavior of function point analysis*, Information and Software Technology. 40, 59-68
[6.] J.J. Dolado (1997), *A Study of the Relationships among Albrecht and Mark II Function Points*, Lines of Code 4GL and Effort, Journal Systems Software, 37, 161-173
[7.] ISO/IEC 14143-1:1998, *Functional size measurement*, ISO standard, 1998
[8.] T. Fetcke (1999), *A Generalized Structure for FPA*, IWSM'99
[9.] H. Diab, M. Frappier, R. St-Denis (2001), *A Formal Definition of COSMIC-FFP for Automated Measurement of ROOM Specification*, Proc. Fourth European Conf. Soft. Measurement and ICT Control, pp. 185-196
[10.] *COSMIC-FFP Measurement Manual*, version 2.2, January 2003
[11.] P.R. Hill (2001), *Practical Project Estimation*, ISBSG

Type Systems for Concurrent Programming Calculi

N. Raja and R.K. Shyamasundar
School of Technology & Computer Science
Tata Institute of Fundamental Research
Mumbai 400 005, INDIA
Email: {raja, shyam}@tifr.res.in

Keywords: Type theory, concurrency, process calculi

Received: March 12, 2003

We explore the role of types *in models of concurrent computation, particularly in the concrete setting of the asynchronous π-calculus. The major theme of this work may be summarized by the slogan* – "Wherever you see structure, think of *types*". *We propose* type *annotations not merely to channels, but also to the highly structured set of processes. The type system guarantees that well typed expressions cannot go wrong. Polymorphic process types formalize extant informal ideas regarding the channel passing and process passing approaches to process mobility. Further, subtyping relation between process types distinguishes between true concurrency and nondeterministic choice.*

1 Introduction

Type systems for sequential programming languages lead to many advantages [3, 20, 27]. In programming practice: types help in structuring programs, they assist in compile-time error detection, and they are useful in optimizing the target code during the compiling process. In the theoretical study of programming language concepts: types help in the creation of succinct metalanguages that act as models for the study of real-life programming languages, and they serve as intermediate code in the task of providing mathematical semantics for programming languages. All this has naturally led to investigations regarding the role of types in theories of concurrency.

The goal of this paper is to examine whether there are any benefits to be gained by introducing *types* in models for concurrent computation. In this paper, we illustrate the role of *types* in the concrete setting of the asynchronous π-calculus (API) [17, 6]. We choose API as it is one of the most prominent calculi for concurrency and communication. API has two kinds of entities – *names* (also called *channels*) and *processes* (also called *agents*). *Names* do not possess any structure, whereas a good amount of structure is needed to build *processes*. The type system we propose, assigns types to both processes and channels. The type assigned to channels, characterizes the length and the nature of the elements that the channel may carry in a communication. The type assigned to processes, characterizes the set of actions that the process is committed to. This results in a rich notion of types which is very useful in the monadic as well as the polyadic versions of API. The type system proposed shows that there are substantial benefits to be reaped by exploring the idea of typing processes. The usage of our type system entails the following advantages:

– It provides a scaffolding for the structured use of the π-calculus, by which we can abolish certain undesirable features – like infinite concurrent activity – right at the early stage of building process terms, rather than at the stage of the reduction system.

– Guarantees *safety*, that well typed expressions will not go wrong.

– Does not constrain the expressive power of the π-calculus.

– Our type system, with minor changes, can be applied to all process algebra formalisms of concurrency. Thus, it provides a uniform basis for the relative assessment of various formalisms. For example, polymorphism in process types brings out potential impredicativity in the semantics of some of these formalisms.

– Subtyping relation among process types helps in distinguishing true concurrency from nondeterministic choice.

The rest of this paper is organized as follows: Section 2 gives a brief review of the asynchronous π-calculus (API); Section 3 presents the type system; Section 4 shows that the type system preserves the semantics of API; Section 5 examines the type system with regard to those properties which are normally of interest in sequential languages; Section 6 explores further extensions to the type system – polymorphism and subtyping – by analogy with traditional type theory; Section 7 describes related work on concurrency and types. The conclusions and future research directions are presented in Section 8.

2 The Asynchronous π-Calculus

In this section, we include a brief review of the asynchronous π-calculus (API) [17, 6] notions that are required for this paper.

Following Milner's idea, a number of calculi for concurrent computation have been proposed, where the communication mechanisms are similar. Communication consists in synchronously sending and receiving through a shared labeled channel.

API [17, 6, 22, 25, 23, 35] is a model of concurrent computation that supports process mobility by naming and passing channels. It consciously forbids the transmission of processes as messages. One of its goals is to demonstrate that in some sense it is sufficiently powerful to allow only names to be the content of communications.

API has two kinds of entities – names (channels), and processes (agents).

Names $(x, y, \ldots \in \mathcal{X})$, have no structure.

Processes $(P, Q, \ldots \in \mathcal{P})$ possess a well defined structure given by

$P ::= 0 \mid \overline{x}y \mid x(y).P \mid P|Q \mid\ !P \mid (\nu x)P \mid$ ERROR

The construct $\overline{x}y$ outputs the name y along x, and does not bind y. The construct $x(y)$ inputs a name, say y, along x, and binds y in the prefixed process. The word 'asynchrony' in this calculus means that message output is non-blocking. This is ensured by restricting the formation of a term $\overline{x}y.P$ in the π-calculus to the case where P is an inactive process. API is powerful enough to encode the synchronous message passing discipline of the π-calculus [36, 30]. The term 0 represents an inactive process. We have extended the π-calculus by including a constant process called ERROR, to represent the kind of type mismatches that we wish to avoid at run-time. The form $P|Q$ means that P and Q are concurrently active, are independent, and can also communicate. The operator "!" is called replication, and $!P$ means $P|P|\ldots$; as many copies as you wish. Finally, $(\nu x)P$ restricts the use of *name* x to P. Apart from input prefix, "ν" is another mechanism for binding names within a process term in API. The operator "ν" may also be thought of as creating new channels.

The operational semantics of API is given in two stages, as shown in Figure 1. A structural congruence is first defined over the process terms, and then a reduction relation is defined. Notice that the rules do not allow reduction under prefix or replication. Also, as expected there are no reduction rules for ERROR. For more details about API, the reader is referred to [17, 6].

3 The Type System

We present our type system in three stages – first, the syntax; second, the typing rules corresponding to API process constructors; and finally, the typing rules corresponding to the reduction system of API. The following subsections are devoted to each of these three stages respectively. Though we use the monadic asynchronous π-calculus to illustrate our typing system, our results can be extended to the polyadic case in a straightforward manner.

3.1 Syntax for Types

We shall call the type information assigned to names as *sort* (ranged over by the metavariable s), and shall use the term *type* (ranged over by the metavariable t) to designate the type information assigned to processes. Our typing scheme is an implicit one (Curry-style typing), because we want to illustrate our work in the setting of a familiar calculus, without any syntactic modifications to the term structure of the calculus.

The sort 's' denotes the length and nature of names which a given channel may carry in a communication. The superscripts R, S, indicate that the channel usage as "receive mode" and "send mode" respectively.

In API, processes may be viewed as programs which manipulate names (which in turn can be considered as data). As mentioned earlier the data manipulated by API programs are unstructured entities. The data develops some structure only in the polyadic extension of API. In the monadic case, the data are atomic entities while in the polyadic case they are n-tuples. Thus the notion of sorts starts making sense only in the polyadic case.

On the other hand, processes have a well-formed structure even in the monadic case; hence types are of significance in both versions of API. The type 't' denotes a process type; it comes in various forms as depicted in Figure 2. The *arrow type* arises due to the *prefix* constructor; the *intersection type* arises due to *par*; and the *recursive type* arises due to *Bang*; the *internal* and *external* types arise due to the *hiding* operator. The API expressions leading to the above types will become clear as we look at each of the typing rules given in the following subsections.

3.2 Types for Processes

An API process $\alpha.A$ can be regarded as an action α and a continuation A. $\alpha.A$ is called a commitment – it is a process committed to act at α [22]. This is precisely the information that the type associated with a process embodies.

Proposition 3.1 (Process-types and Commitment) *A process type describes the sequence of actions that a process is committed to.*

This will become clear from the following subsections.

API is based on the object model of computing [26]. Objects have an independent identity and they have a persistent state which may not be entirely visible to the other agents. Thus the type associated with a process has two facets – one which specifies its interface on the outside and the other which determines its internal transitions. Our type system brings out this aspect of API explicitly by making

Definition 2.1 (Structural Congruence over Process Terms)
$\equiv$ is the smallest congruence relation over process terms such that the following laws hold:

1. Processes are identified if they only differ by a change of bound names
2. $(\mathcal{P}/\equiv, |, 0)$ is a symmetric monoid
3. $!P \equiv P|!P$
4. $(\nu x)0 \equiv 0, (\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$
5. If $x \notin freeNames(P)$ then $(\nu x)(P|Q) \equiv P|(\nu x)Q$
6. $P|\text{ERROR} \equiv !\text{ERROR} \equiv (\nu x)\text{ERROR} \equiv \text{ERROR}$

Definition 2.2 (Reduction Relation)
The reduction relation $\rightarrow$ over processes is the smallest relation satisfying the following rules:

$$\text{Comm} \quad (\ldots + x(y).P) \mid (\ldots + \overline{x}[z].0) \;\rightarrow\; P\{y \leftarrow z\} \mid 0$$

$$\text{Par} \quad \frac{P \rightarrow P'}{(P|Q) \rightarrow (P'|Q)}$$

$$\text{Struct} \quad \frac{Q \equiv P \quad P \rightarrow P' \quad P' \equiv Q'}{Q \rightarrow Q'}$$

$$\text{Res} \quad \frac{P \rightarrow P'}{(\nu x)P \rightarrow (\nu x)P'}$$

Figure 1: Operational Semantics of API

Sorts	$s ::= BasicSort \mid (s)^R \mid (s)^S$
Type − Variables	$T \mid U \mid V$
Pre − Types	$\sigma ::= \epsilon \mid \phi \mid T \mid Name(Name : s)^R \mid Name(Name : s)^S \mid \sigma \rightarrow \sigma \mid \sigma \cap \sigma \mid \mu T.\sigma$
Pre − Types	$\sigma \mid \sigma_{ext} \mid \sigma_{int}$
Types	$t ::= < \sigma_{ext}, \sigma_{int} >$
TypeEnvironments	$\Gamma ::= \{\} \mid \Gamma, x : s \mid \Gamma, P : t$

Figure 2: Syntax of the Type System

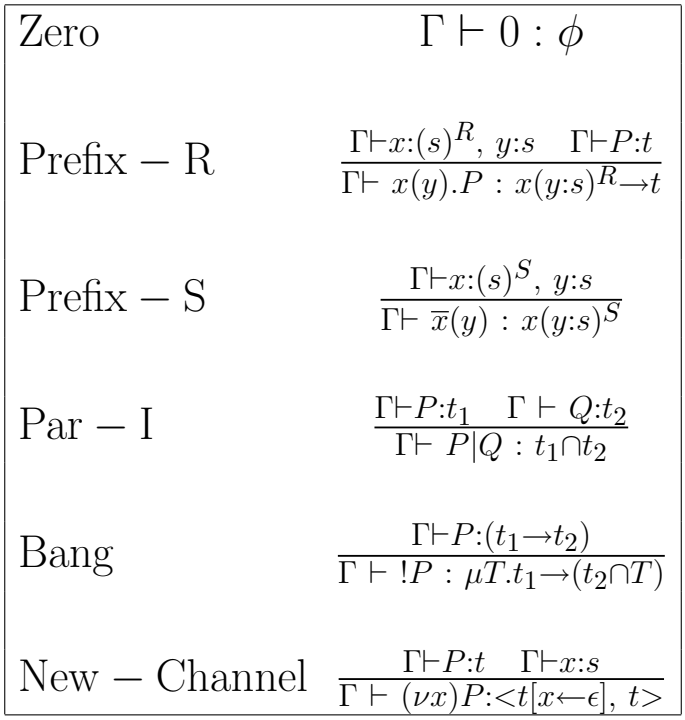


Figure 3: Typing Rules for Process Constructors

the type associated with a process to be a tuple comprising its external and internal types respectively.

The typing rules corresponding to each of the process constructors that API allows, are listed in Figure 3. Among all the typing rules listed in Figure 3, the internal and external types turn out to be distinct only when the 'hiding operator' occurs in the process term. Hence, only the *New-Channel* typing rule shows both components of the *type* associated with a process term. The types are to be viewed as being implicitly universally quantified on name sorts. The typing rules are given in a syntax directed way, and can be checked for well-formedness by structural induction over the API syntax.

Arrow types

Arrow types are familiar from type systems for sequential programming. The typing rule *Prefix-R* states in its premises that if x, y are names, y has *sort* s, and x has sort $(s)^R$ – which means that the channel x may be used for receiving a *name* of sort s – and the process P has type *t*; then the API term $x(y).P$ is assigned the type $x(y : s)^R \rightarrow t$. The type indicates that process $x(y).P$ can use channel x for receiving only, indicated by the superscript R. Further, after such a communication occurs (and only after), it may proceed to behave like a process having type t. This strict sequentiality imposed by the prefix constructor of API is made explicit by the $\rightarrow$. The rule *Prefix-S* is very similar except that it shows that the name x may be used only for sending (the superscript S) by the newly constructed process.

We shall discuss the prefix rule again when we consider higher-order models for concurrency. The *Prefix* type rules will reveal any impredicativity which could be lurking in the semantics of the calculus being typed. More about impredicativity will be discussed in Section 7.

Intersection types

The rule *Par-I* says that the *intersection type* '$\cap$' arises when a process is built by the parallel composition of two other process terms. The parallel composition operator '|' allows the components to make transitions independently (i.e., disjoint parallelism). Thus, the set of actions that a process belonging to an intersection type can indulge in, is given by the conjunction of the set of possible actions of its component processes. Intersection types are also called 'conjunctive types' in the parlance of type theory.

There is a notable difference between the conventional usage of intersection types [3], and the way they are used in this work. In this work, the intersection type corresponds to a process constructor (par, '|'). Traditionally, intersection types are used for typing a term which belongs to various structurally unrelated types. For example, the symbol '+' is used to represent integer addition, and real addition. The type assigned to such a function is $((int \rightarrow int \rightarrow int) \cap (real \rightarrow real \rightarrow real))$. In other words, conventional intersection types are used to represent 'overloading'. Notably also absent from our type system, is the universal type ω (such that $P : \omega$ for all terms P), which accompanies intersection types normally.

The parallel composition operator '|' also allows the

components to communicate. Hence we shall encounter the *intersection type* '$\cap$' once again in the typing rule describing communication between the two component processes.

Recursive Types

The *Bang* typing rule is another instance where the relevance of types in concurrency is very clearly brought out. The operator "!" is called replication and $!P$ – "bang P"– means $P|P\ldots$; as many copies as you wish. In API the "!" operator can be applied to any process term P to form the process $!P$ (where P has been constructed using any rule for building processes). The important point to be noted is that API does not enforce any restrictions on P before the "!" operator may be applied to it.

However the typing rule *Bang* states in its premise that the type of P should be an "arrow type" such as $(t_1 \rightarrow t_2)$ before we can apply "!" to P to get $!P$. This makes it mandatory that the outermost constructor of process P be a prefix, before the "!" may be applied to it. Thus the replication operator can be used on guarded processes only. $!\pi.P$ is a common instance of replication – it indicates a resource P which can be replicated only when a requester communicates via π. This shows that the premise in the typing rule *Bang* is meaningful. The next question which arises is whether the typing rule *Bang* is being too restrictive by imposing such a condition. Before we answer this query, let us examine the meaning of a term such as $!P$ when it is not required of P that its outermost constructor be a prefix. Such a term, "$!P$", means a resource which replicates asynchronously – replicates without demand, without requirement. $!P$ appears to be acting on its own *free will*, so to say. In other words it represents *infinite concurrent activity*. Now this is certainly not a meaningful construct, and we would rather not have such a term in our calculus. Hence the typing rule *Bang* does not strip API off any expressive power; in fact it rules out an entire class of meaningless terms from being constructed. API abolishes such behaviour by taking recourse to its reduction rules. However we have done better in our type system, in that, we even forbid the occurrence of such terms right at the level of syntax, by enforcing a discipline in the structured construction of API programs.

After having looked at the premise, let us now examine the conclusion of the *Bang* rule. It infers that the process term $!P$ has the type $(\mu T.t_1 \rightarrow (t_2 \cap T))$. μ represents recursion and the type variable T is the parameter of the recursion. The recursive type makes the recursive behavior of "!" operator explicit. The intuition provided by the recursive type is well supported when we turn to API and find that all parametric recursive process definitions can be encoded by replication. Let us come back to the *Bang* typing rule: When P has the type $(t_1 \rightarrow t_2)$, it means that P behaves as dictated by the type t_1 and then (sequentially) behaves as dictated by t_2. The recursive type assigned to $!P$ says that $!P$ behaves as required by t_1 and then as required by $(t_2 \cap T)$. The intersection type mirrors the fact that an independent process of type t_2 has been spawned, which executes in parallel with the resource of type T. But T is the parameter of recursion, and we eliminate it by recursive unfolding, that is we replace T by $(\mu T.t_1 \rightarrow (t_2 \cap T))$ and proceed further as before.

The Recursive type in this setting is very similar to that used in sequential programming. The type $\mu T.t_p$ stands for the least fixed point solution of the type equation $T = t_p$. The solutions of such equations will be infinite types, which can be represented by infinite labeled binary trees. The definition of such trees is provided in Figure 4. The same Figure also gives a congruence relation on types with the help of such trees [9, 10].

Internal and External Types

In all the typing rules that we have considered so far, the external and internal types are identical. However, the operator ν used as $(\nu x)P$ localizes (restricts) the use of the channel x within P. The channel name x is guaranteed to be different from any other channel name which finds an occurrence outside P. Hence communications can be sent and received on x only internally within process P. This brings us to the next typing rule, *New-Channel*, which gives the external and internal type of a process term which has been built using the operator ν. The notion of distinguishing between the external and internal type of a process is derived from the notion of existential types and explicit witnesses [28], and the notion of partially abstract types [8]. The *external* type-component states that if the process P has type t and the channel x has sort s, then the external type of the process term $(\nu x)P$ is $t[x \leftarrow \epsilon]$ which means that in the type t all occurrences of x are replaced by ϵ, thereby making the channel x unavailable for communication with the outside world. The *internal* type-component states that as far as the internal type of $(\nu x)P$ is concerned, there is no change, the type continues to be t.

That explains all the typing rules that have arisen because of the process constructors that are allowed in API.

3.3 Reduction rules and Types

The typing rules shown in Figure 4 correspond to the congruence relation over types. They spell out when two types may be considered to be congruent.

The remaining typing rules, shown in Figure 5, correspond to the reduction system of API. The typing rules *Inter-E*, *Comm*, *Par-R*, *Res*, and *Struct* tell us how to consistently infer the type of the term which results from a reduction.

The rule *Comm* mentions the types required of each term so that the communication between the two processes will result in a proper reduction (one which does not result in ERROR), and gives the type of the resultant process. The rule *Par-I* mentioned earlier as giving rise to the *intersection type* '$\cap$', can be considered to be a special case of this rule. If there is no communication possibility allowed by

Definition 3.2
The tree corresponding to the process type t, written as $T(t)$, is defined as follows:

$T(\phi) = \phi$;

$T(t_1 \rightarrow t_2) = (\rightarrow, T(t_1), T(t_2))$;

$T(t_1 \cap t_2 = (\cap, T(t_1), T(t_2))$;

$T(\mu T.\, t) = T(t[T \leftarrow \mu T.\, t])$.

Definition 3.3
$\approx_t$ is the smallest congruence relation over types, such that, the following laws hold:

CR-1 Process types are identified if they only differ by a change of bound names;

CR-2 $t \cap \phi \approx_t \phi \cap t \approx_t t$;

CR-3 $t_1 \approx_t t_2$, if $T(t_1) = T(t_2)$.

Figure 4: Congruence Relation for Types

$$\text{Inter} - \text{E} \quad \frac{\Gamma \vdash P{:}t_1 \cap t_2}{\Gamma \vdash P{:}t_1 \qquad \Gamma \vdash P{:}t_2}$$

$$\text{Comm} \quad \frac{x(y).P : (x(y{:}s)^R \rightarrow t_P), \quad \overline{x}(z) : (x(z{:}s)^S)}{x(y).P \mid \overline{x}(z) \rightarrow P\{y \leftarrow z\} : t_P\{y \leftarrow z\}}$$

$$\text{Par} - \text{R} \quad \frac{P{:}t_P \rightarrow P'{:}t_{P'}}{(P|Q) : t_P \cap t_Q \rightarrow (P'|Q) : t_{P'} \cap t_Q}$$

$$\text{Res} \quad \frac{P{:}<t_P, t_P> \rightarrow P'{:}<t_{P'}, t_{P'}>}{(\nu x)P{:}<t_P[x \leftarrow \epsilon], t_P> \rightarrow (\nu x)P'{:}<t_{P'}[x \leftarrow \epsilon], t_{P'}>}$$

$$\text{Struct} \quad \frac{t_Q \equiv_t t_P \quad P{:}t_P \rightarrow P'{:}t_{P'} \quad t_{P'} \equiv_t t_{Q'}}{Q{:}t_Q \rightarrow Q'{:}t_{Q'}}$$

Figure 5: Reduction Rules and Types

the types of the interacting processes (disjoint parallelism), then the resulting type of the compound term is given by the *Par-I* typing rule. It is worth noting that the typing rules corresponding to process constructors and the typing rules corresponding to the reduction system, cannot be kept separated in the type system for API. This is because the operator *par* '|' is overloaded – it represents both concurrency (a process building operation), as well as communication (an operation which is a part of the reduction rules). However, such a clear separation can be achieved in the case of a type system constructed along similar lines for Boudol's *concurrent λ-calculus* [5].

Once again we mention that in all these rules, except *Res* the inference is valid for both components of the process type – external as well as internal. The *Res* typing rule explicitly indicates the process type as a tuple and gives the corresponding new components of the type after reduction.

4 Soundness and Type Safety

In this section, we examine the effect of the type-system on the semantics of API. First, we show that our type system preserves the semantics of API, and prove that well typed expressions never reduce to ERROR – which means process types guarantee the safety property.

The operational semantics of API was defined in two stages [22, 26] as shown in Section 2. A structural equivalence on process terms was given first, and then a reduction relation was given which describes the act of communication. We prove below that our notion of *type* is consistent with each of these two stages.

Theorem 4.1 *Types preserve the structural congruence rules on process terms.*

Proof: We prove this theorem by examining the structure of the definition of structural congruence on process terms.

1. Types respect α-conversion (typing rule *CR-1*), hence agents are identified if they only differ by a change of bound names.

2. Using the typing scheme presented in this paper, we show that types preserve the fact that $(\mathcal{P}/\equiv, |, 0)$ is a symmetric monoid.
   $$0 : \phi \quad (Zero)$$
   $$P|0 : t \cap \phi \quad (Par-I)$$
   $$t \cap \phi \approx_t t \quad (CR-2)$$
   Similarly,
   $$0|P : \phi \cap t \quad (Par-I)$$
   $$\phi \cap t \approx_t t \quad (CR-2)$$
   By steps 3 and 5, it follows that types preserve the monoidal structure of $P/\equiv$, where '|' is the associative operator of the monoid, and 0 forms the identity w.r.t '|'.

3. The typing rule *Bang* has been explained in sufficient detail in Section 3. It clearly follows from the illustration given there that types guarantee $!P \equiv P|!P$.

4. The inactive process 0 has the type ϕ as both its external and internal type. The restricted process $(\nu x)0$ continues to have the same type. Hence $(\nu x)0 \equiv 0$.
   If the process P has the process type $< E_P, I_P >$ then the process term $(\nu x)(\nu y)P$ has the type $< E_P[x \leftarrow \epsilon, y \leftarrow \epsilon], I_P >$ which is equivalent to the type $< E_P[y \leftarrow \epsilon, x \leftarrow \epsilon], I_P >$ associated with the process term $(\nu y)(\nu x)P$.

5. From the typing rules *Par-I*, and *New-Channel* it immediately follows that if x is not free in P then $(\nu x)(P|Q) \equiv P|(\nu x)Q$.

Thus *types* preserve the structural congruence on process terms. □

Theorem 4.2 *Well typed expressions can never reduce to* ERROR.

Proof: In the absence of types, the reduction rule which allows communication between process terms states that $x(y).P \mid \overline{x}z \rightarrow P\{y \leftarrow z\}$. The typing scheme assigns to each of the two concurrent process terms the following types –
$x(y).P : x(y : s)^R \rightarrow t_P$, and $\overline{x}(z) : x(z : s)^S$
Further the type scheme allows a reduction to take place by the typing rule *Comm* only when the two types are complementary and the sorts of the channels being used for communication are consistent with each other. These are exactly the conditions required to ensure a meaningful reduction in the π-calculus. The term resulting from the communication is $P\{y \leftarrow z\}$ and its corresponding type is $t_P[y \leftarrow z]$. Then well typed process terms never reduce to ERROR. □

Theorem 4.3 *The type system preserves the semantics of* API.

Proof: Follows as a direct consequence of Theorem 4.1 and Theorem 4.2. □

5 Basic Syntactic Properties

The type system proposed in this work is meant for concurrent calculi, and as is well known, the requirements of concurrent systems are quite different from those of sequential systems. However, there are a number of syntactic properties which have been of interest in traditional type systems for sequential programming [3]. For the sake of completeness we briefly examine such properties in our type system.

1. **Implicit Typing:** The typing scheme we have proposed is an implicit one (Curry-style typing). We chose Curry-style typing because we wanted to illustrate our work in the setting of a familiar calculus without requiring major syntactic modifications to the term structure of the calculus.

2. **Church-Rosser Property (CR):** This is more a property of the underlying calculus being typed, rather

than the type system itself. In our case, API does not satisfy the Church-Rosser property, since functions such as 'parallel-or' can be represented in it.

3. **Subject Reduction (SR):** If process term P has the type t_P, and if P reduces to the term P'; then the subject reduction property states that the type of P' is also t_P. Such a property does not hold in our type system because process reduction in API is non-deterministic, and also due to name passing, the interface of a process may change with reduction.

4. **Strong Normalization (SN):** This property states that all reduction sequences terminate eventually. This means that not every computable function is definable in the system. However this property does not hold in our type system because of the presence of recursive process types. With the help of recursive process types we are able to type the "!" operator of API without restricting its expressive power.

5. **Type Checking:** This property states whether, given a typing environment Γ, a process term P, and a type t, is the judgment $\Gamma \vdash P : t$ decidable or not. Type checking is decidable for our type system.

6. **Type Inference:** This requires that given Γ and P, it should be possible to compute a t such that $\Gamma \vdash P : t$ is valid. Type inference is possible for process types.

The above properties gained prominence because of their importance in the traditional application areas of types, such as in proof theory and in sequential programming. In the domain of concurrency, many of the above properties such as CR, SR, and SN are no longer relevant. Instead, properties such as *safety* and *liveness* become important.

6 Further Extensions to the Type System

There are a number of concepts which have played a significant role in the success of type disciplines for sequential systems. Two such concepts are *Polymorphism* and *Subtyping*. In this section we examine whether these concepts shed any light on concurrent calculi. We informally extend our type discipline in two directions – to incorporate polymorphism and subtyping. The results are indeed very promising as we demonstrate in the following subsections. Further research along these lives is sure to lead to insights into concurrent calculi.

6.1 Channel passing versus Process passing

Many distinct formalisms [25, 29, 37, 1, 18, 2, 5] have been invented to describe systems which do not have fixed interconnection topology between processes. All such formulations may be classified into two groups by examining the way in which they achieve mobility. One group achieves mobility by allowing channel names to be communicated [25, 1, 18] – the π-calculus belongs to this group. The other group achieves mobility by supporting the transmission of processes as messages [37, 2, 29, 5] – let us take a particular example from this group, say CHOCS [37].

The name passing approaches to concurrency allow names, but not processes, to be transmitted in communications. On the other hand, the process passing approaches allow processes, but not names, to be transmitted as messages. There are relevant reasons why each of these two approaches allows only either names or processes but not both to be the content of communications. Thus neither of the two approaches can be said to have achieved "uniformity" in dealing with their primitive entities. Further it has been demonstrated [37, 22, 36] that both the paradigms are equally powerful as far as their expressive power is concerned.

The question that we ask now is whether our type system can provide any relevant criteria that favours the choice of one paradigm over the other? The answer is in the affirmative – the type system does provide a measure which helps in discriminating the two paradigms.

In order to see how, let us examine the type that our system assigns to the process constructor which allows abstraction of names and processes in the paradigms of name-passing and process-passing calculi respectively. In this section, let x, y range over *Names*; P, Q range over *Processes*; N_s range over *Name Sorts*; P_t and t_q range over *Process Types*.

Consider the following π-calculus term, and its corresponding type – $x(y).Q : \forall N_s.x(y : N_s)^R \rightarrow t_q$. The type expression states that the process term $x(y).Q$ behaves like a program which expects any name y as input (y is a dummy parameter), and then behaves like the process Q. However there is no restriction on what sort of name it can accept as input, as shown by the universal quantifier which ranges over N_s. The important point to be observed is that the entity "$\forall N_s.x(y : N_s)^R \rightarrow t_q$" is itself a process type and does not lie in the range of the universal quantifier (which ranges only over name sorts in this case).

Now consider the following CHOCS term, and its corresponding type – $x?(P).Q : \forall P_t.x(P : P_t)^R \rightarrow t_Q$. In this case the type expression states that the process term $x?(P).Q$ behaves like a program which expects any P process as input (P is a dummy parameter), and then behaves like the process Q. However the program does not impose any restrictions on the type of the input process (represented by the universal quantifier ranging over P_t. In this case the entity "$\forall P_t.x(P : P_t)^R \rightarrow t_Q$" is itself a process type and hence the universal quantifier ranges over this type as well. In other words process types turn out to be impredicative in CHOCS, while they remain predicative in the π-calculus.

It is a well known phenomenon in type theory that the semantics of a predicative formalism is extremely simple and elegant in comparison with the semantics required by an impredicative formalism [11]. Thus conceptual simplicity

and elegance in the semantics of the type system associated with a formalism favours π-calculus over CHOCS – or in more general terms, name passing approaches over process passing approaches to concurrency.

6.2 True Concurrency versus Nondeterministic Interleaving

As mentioned in Section 7, the work by Pierce and Sangiorgi has shown that the subtyping relation among name sorts leads to an interesting refinement. In this subsection we examine the relevance of subtyping relation among process types.

In the semantic theories for process algebras such as CCS [21] and CSP [14], concurrency is semantically reduced to nondeterminism. For example the process $a|b$ is considered semantically equivalent to the process $(a.b + b.a)$. It has been demonstrated by Boudol et al. [7], that in certain situations it is meaningful to retain concurrency as a primitive concept without reducing it to nondeterministic interleaving. We now show that process types can be used to maintain such a distinction.

For this purpose we introduce *union types*, '$\cup$', and a *subtyping* relation among union types. Consider a process term of the form $P + Q$. This term can (nondeterministically) indulge, either in the actions specified by P or in the actions specified by Q (exclusive-or of the actions). If the types of P and Q are given by t_p and t_q respectively, then we assign to the process $P + Q$, the type $t_p \cup t_q$. Now we define the subtyping relation '$\subseteq$', by the relations, $t_p \subseteq (t_p \cup t_q)$ and $t_q \subseteq (t_p \cup t_q)$. The *subtyping* relation is reflexive, antisymmetric, and transitive. Intuitively in a context which requires an object of type t, one could as well use an object whose type is a subtype of t, but not vice versa. This intuition is well supported when we examine the process terms themselves. It is important to note that such a subtyping relationship does not hold in the case of intersection types i.e. $t_p \not\subseteq (t_p \cap tq)$ and $t_q \not\subseteq (t_p \cap t_q)$.

Thus we get the type of $a|b$ as $(t_a \cap t_b)$ and the type of $((a.b) + (b.a))$ as $((t_a \rightarrow t_b) \cup (t_b \rightarrow t_a))$. Consider the above processes after they make a transition on 'a'. $(a.b) + (b.a)$ reduces to b. The new process type is a subtype of the original process type, i.e. $t_b \subseteq ((t_a \rightarrow t_b) \cup (t_b \rightarrow t_a))$. On the other hand the process $a|b$ also reduces to b. But the distinction lies in the fact that the new process type is not a subtype of the original process type, i.e. $t_b \not\subseteq (t_a \cap t_b)$. Thus the type equivalence provided by the subtype relation provides a key to distinguish true concurrency from nondeterministic interleaving.

7 Related Work

In this section we briefly discuss work related to type systems for mobile processes. As mentioned earlier, the concurrent calculi that were proposed following Milner's CCS, have two basic syntactic entities – *channels* and *processes*. This situation is unlike that in sequential programming, where the λ-calculus (the de-facto standard sequential language), has only one basic entity – *terms*. Till now a major part of the research on type systems for concurrency has concentrated on assigning type information to the channels only. Such type information has been called *sorts*.

The relevant starting point is the notion of *sorts* introduced in the polyadic π-calculus by Milner [22]. We illustrate Milner's notion of *sorting* with an example. Consider the process term $\overline{x}y.0|x(u).\overline{u}().0|\overline{x}z.0$. In this expression, channels y and z carry only the empty vector if they are ever used for communication. On the other hand, channel x always carries another channel name, which in turn is used in communicating an empty vector. We can represent these observations as: $\{y \mapsto (), z \mapsto (), x \mapsto (())\}$. Notice that the usage of x is characterized by a nesting of parentheses. The above representation is precisely the *sorting* as proposed by Milner. Thus the *sort* associated with a channel captures the length and nature of the vector that the name carries in communications. In the polyadic π-calculus, names may carry n-tuples of other names. Hence the notion of sort information assumes prominence only in the polyadic setting. There are some more points to be noted. Firstly, sort information is assigned to channels only and sort equivalence is by *name matching*. Secondly, names occurring in a perfectly meaningful π-calculus process term may not have any *sorting* at all. This can occur if a term uses names to communicate different entities at different times. Thus the lack of a proper *sorting* does not render a π-calculus expression meaningless. Finally, *sorts* are implicit i.e., they do not occur in the term structure of the calculus. Honda [15] presented similar results, in an independent work. Gay [12] presents an algorithm (quadratic in the length of the input process) for automatically inferring such sort information for channels, from the given π-calculus term. Naturally *sorts* are inferred only if they exist. Honda and Vasconcelos [16] gave an algorithm to the same effect, though linear in the size of the input process. Following Lafont's work on interaction nets [19], Honda proposed conditions on channel sorts, so as to achieve freedom from deadlock in certain finite and simple situations.

Pierce and Sangiorgi [31] extended the notion of sorts by distinguishing between the ability to read from a channel, the ability to write to a channel, and the ability to do both. This refinement gives rise to an interesting subtype relation on channel sorts. Their sort equivalence is by *structural matching*. In Pierce's work, sorts appear explicitly in the term structure and further such sort information is even communicated from one process to another. This requires changes in the π-calculus model, thus resulting in a different concurrent calculus. In Pierce's work, the problem of algorithmic inference of sort information is not considered at all.

The idea of assigning type information to processes has also been used by researchers in other contexts [29, 13, 34]. In Facile, CML, and the Typed λ-calculus with first class processes, the notion of process type is present. However

the process types which find usage in these programming languages are predominantly just functional types. The notion of polymorphism has been included in Facile and CML, but once again in the realm of channel sorts, in order to derive more flexible sorting mechanisms.

From the above observations it is clear that the notions of type inference, polymorphism, subtyping, and conditions for deadlock freedom have been explored in the domain of channel sorts. Such investigations in the domain of process types, would yield rich dividends [33, 4, 32, 38, 39].

8 Conclusions and Future Directions

The aim of this work was to establish a bridge between the disciplines of concurrency and type theory. We presented a novel operational semantics for the asynchronous π-calculus, by making reductions sensitive to type. Our type system was unique, in not confining type information to channels only; very informative types were assigned to processes also. The universe of process terms with its rich structure, proved to be a fertile ground for the application of various type constructors. The type system did not restrict the expressive power of the asynchronous π-calculus in any way. Types guaranteed *safety*, that well typed expressions would not go wrong. Further the type system helped in preventing the construction of meaningless expressions, such as those representing infinite concurrent activity, right at the stage of syntactic formation of process terms. The notion of polymorphism brought out the latent impredicativity in the semantics of the process-passing approaches to concurrency. The notion of subtyping helped in distinguishing true concurrency from nondeterministic interleaving.

As further work, it would be highly interesting and relevant to explore how the notion of process types could be put to use in reasoning about *liveness* properties of concurrent systems, such as freedom from deadlock. It would also be fruitful to pursue work towards establishing algebraic equivalences over process types. Also as discussed in the last section, exploring the notions of polymorphism and subtyping looks promising.

This work is part of an ongoing investigation into the role of type theoretic concepts in the setting of concurrency. It would also be productive to carry out such an investigation in a more abstract formalism for concurrency, e.g., like the one provided by *action structures* [24].

Acknowledgment

We wish to thank the anonymous referees for constructive comments which were of help in improving the content and presentation of this paper. Our thanks to Ms. Margaret D'Souza for typing and typesetting this paper.

References

[1] E. Astesiano, and G. Reggio (1984) Parametric Channels via Label Expressions in CCS, *Theor. Comp. Science*, Vol. 33, pp. 45–64.

[2] E. Astesiano and G. Reggio (1987) SMoLCS-driven concurrent calculi, *Lecture Notes in Computer Science*, Springer-Verlag, Vol. 249, pp. 169–201.

[3] H. Barendregt, and K. Hemerik (1990) Types in lambda calculi and programming languages, *Proc. ESOP'90*, LNCS 432, pp. 1–36.

[4] M. Berger, K. Honda, and N. Yoshida (2003) Genericity and the π-Calculus, *Proc. FOSSACS'03*, LNCS, To appear.

[5] G. Boudol (1989) Towards a lambda-calculus for concurrent and communicating systems, *Proc. TAPSOFT'89*, LNCS 351, Springer-Verlag, pp. 149–161.

[6] G. Boudol (1992) Asynchrony and the π-calculus, *Rapport de Recherche*, Number 1702, INRIA Sophia-Antipolis.

[7] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn (1991) Observing Localities, *INRIA Report No. 1485*.

[8] L. Cardelli, and P. Wegner (1985) Understanding Types, Data Abstraction, and Polymorphism, *ACM Computing Surveys*, Vol. 17 (4).

[9] F. Cardone, and M. Coppo (1990) Two Extensions of Curry's Type Inference System, *Logic and Computer Science*, Academic Press, pp. 19–75.

[10] B. Courcelle (1983) Fundamental Properties of Infinite Trees, *Theoretical Computer Science*, Vol. 25, pp. 95–169.

[11] R.L. Constable (1991) Type Theory as a Foundation for Computer Science, *Proc. TACS'91*, Lecture Notes in Computer Science, Vol. 526, Springer-Verlag.

[12] S. Gay (1993) A sort inference algorithm for the polyadic π-calculus, *Proc. ACM Symposium on Principles of Programming Languages*, ACM Press.

[13] A. Giacolone, P. Mishra, and S. Prasad (1989) Facile: A symmetric integration of concurrent and functional programming, *Int. Jl. of Parallel Prog.*, Vol. 18, pp. 121–160.

[14] C.A.R. Hoare (1985) Communicating Sequential Processes, *Prentice-Hall*, London.

[15] K. Honda (1993) Types for Dyadic Interaction, *Proc. CONCUR'93*, Lecture Notes in Computer Science, Volume 715, Springer-Verlag.

[16] K. Honda, and V.T. Vasconcelos (1993) Principal typing schemes in a polyadic π-calculus, *Proc. CONCUR'93*, LNCS 715, Springer-Verlag.

[17] K. Honda, and M. Tokoro (1991) An Object Calculus for Asynchronous Communication, *ECOOP'91*, Lecture Notes in Computer Science, Volume 512, Springer-Verlag.

[18] S.R. Kennaway and M.R. Sleep (1985) Syntax and informal semantics of DyNe, a parallel language, *LNCS 207*, Springer-Verlag, pp. 222–230.

[19] Y. Lafont (1990) Interaction Nets, *Proc. POPL'90*, ACM Press, pp. 95–108.

[20] B. Mahr (1993) Applications of Type theory, *Proc. TAPSOFT'93*, Lecture Notes in Computer Science, Volume 668, Springer-Verlag.

[21] R. Milner (1989) Communication and Concurrency, *International Series in Computer Science*, Prentice Hall.

[22] R. Milner (1991) The polyadic π-calculus: a tutorial, *Logic and Algebra of Specification*, Proceedings of International NATO Summer School (Marktoberdorf, Germany), Series F, Vol. 94, Springer.

[23] R. Milner (1999) Communicating and Mobile Systems: The Pi Calculus, Cambridge University Press.

[24] R. Milner (1993) Action Structures and the π-Calculus, *Proof and Computation*, Proceedings of International NATO Summer School (Marktoberdorf, Germany), Series F, Vol. 139, Springer.

[25] R. Milner, J. Parrow, and D. Walker (1992) A calculus of mobile processes (Parts I and II), *Information and Computation*, Vol. 100, pp. 1–77.

[26] R. Milner (1992) Functions as processes, *Journal of Mathematical Structures in Computer Science*, Vol. 2 (2), pp. 119–141.

[27] J.C. Mitchell (1990) Type Systems for Programming Languages, *Handbook of Theoretical Computer Science*, Elsevier Science Publishers.

[28] J.C. Mitchell, and G. Plotkin (1988) Abstract Types have Existential Types, *ACM Transactions on Programming Languages and Systems*, Vol. 10 (3).

[29] F. Nielson (1989) The typed λ-calculus with first class processes, *Proc. PARLE'89*, Lecture Notes in Computer Science, Volume 366, Springer-Verlag.

[30] C. Palamidessi (1997) Comparing the expressive power of the Synchronous and the Asynchronous pi-calculus, *Proc. ACM Symposium on Principles of Programming Languages*, ACM Press, pp. 256–265

[31] B. Pierce, and D. Sangiorgi (1993) Typing and Subtyping for Mobile Processes, *Proc. IEEE Symposium on Logic in Computer Science*, IEEE Press.

[32] B. Pierce, and D. Sangiorgi (2000) Behavioral Equivalence in the Polymorphic Pi-Calculus, *Proc. Journal of ACM*, Vol. 47 (3) pp 531–584.

[33] N. Raja, and R.K. Shyamasundar (1994) Type Systems for Concurrent Calculi, *Proc. of the Tenth Workshop on Abstract Data Types* (ADT'94), Santa Margherita Ligure, Genoa, Italy.

[34] J.H. Reppy (1993) Concurrent ML: Design, Application and Semantics, *Funct. Prog., Concurrency, Simulation and Automated Reasoning*, LNCS 693, Springer-Verlag.

[35] D. Sangiorgi, and D. Walker (2001) The Pi-Calculus – A Theory of Mobile Processes, Cambridge University Press.

[36] D. Sangiorgi (1993) From π-calculus to Higher-Order π-calculus — and back, *Proc. TAPSOFT '93*, Lecture Notes in Computer Science, Volume 668, Springer-Verlag.

[37] B. Thomsen (1993) Plain CHOCS. A Second Generation Calculus for Higher Order Processes, *Acta Informatica*, Vol. 30 (1), pp. 1–59.

[38] D.N. Turner (1996) The Polymorphic Pi-Calculus: Theory and Implementation, *Ph.D. Thesis*, University of Edinburgh.

[39] V. Vasconcelos (1994) Typed Concurrent Objects, *Proc. ECOOP'94*, LNCS, Springer-Verlag, pp. 100–117.

Improving Efficiency of Program Graph Scheduling with Partial Strict Triggering of Program Graph Nodes

Milan Ojsteršek and Aleksander Kvas
FERI Maribor, Smetanova 17, Maribor, Slovenia
Phone: +386 2 220 7451, Fax: +386 2 251 1178
E-mail: ojstersek@uni-mb.si

Keywords: parallel computers, program graphs, scheduling

Received: June 15, 2003

An efficient scheduling of a parallel program onto the processors is critical for achieving a high performance from a parallel computer system. The scheduling problem is known to be NP-hard and heuristic algorithms have been proposed to obtain optimal and sub optimal solutions. The partitioning algorithm partitions an application into tasks with appropriate grain size and represents them in the form of a directed acyclic graph (DAG). The nodes of the resulting DAG are then scheduled onto the processors of a parallel computer system. We can see that almost all coarse grained program graph nodes don't need all of their input operands at the beginning of their execution. Thereafter they can be scheduled a bit earlier. This type of program graph nodes triggering is called partial strict triggering. The missing operands will be requested later during the execution. Coarse grained program graph nodes send their output operand to all successors, as soon as they produce them. Successors of coarse grained program graph nodes will be scheduled earlier too, because they will receive their input operands sooner. An evaluation of improved CPM, VL and DSH scheduling algorithms is done in this paper. We have improved them with partial strict triggering of coarse grained program graph nodes.

1 Introduction

Optimal execution of parallel programs which are executed on a parallel computer system depends on partitioning programs into modules and scheduling those modules for the shortest possible execution time. In this paper, we present three efficient algorithms for scheduling modules to the processing units of a parallel computer system. An algorithm for the partitioning programs into modules is discussed in (Ojsteršek 1994). We will make only a brief description of it in chapter 2.

The general problem of multiprocessor scheduling can be stated as scheduling a set of partially ordered computational tasks onto a multiprocessor system so that a set of performance criteria will be optimised. The difficulty of the problem depends heavily on the topology of the program graph representing the precedence relations among the tasks, the topology of the multiprocessor system, the number of parallel processors, the uniformity of the node processing time and the performance criteria chosen. In general, the multiprocessor scheduling problem is computationally intractable even under simplified assumptions. Because of this computational complexity issue, many heuristic algorithms have been proposed to obtain optimal and suboptimal solutions to various scheduling problems (Palis et al. 1996, Gerasoulis & Yang 1992, Darbha & Agrawal 1998, Park & Chloe 2002).

The scheduling of programs onto parallel computer system can be achieved using three approaches: static, dynamic and hybrid. The distinction indicates the time at which the scheduling decisions are made. With *static* scheduling, information regarding the program graph representing the program must be estimated prior to execution. In static scheduling, each node of a program graph has a static assignment to a particular processor, and each time that task is submitted for execution, it is assigned to that processor. In *dynamic* scheduling, the parallel processor system must attempt to schedule tasks on the fly. Thus, the scheduling decisions are made while the program is running. The disadvantage of dynamic scheduling is the overhead incurred to determine the schedule while the program is running. *Hybrid* scheduling technique are a mix of static and dynamic methods, where some preprocessing is done statically to guide the dynamic scheduler and/or reduce the amount of undeterminism.

We have improved three well known static scheduling algorithms (CPM, VL and DSH) with partial strict triggering of program graph nodes. We compare our improved algorithms with original CPM static scheduling algorithm.

This paper is organised as follows. First we give a brief description of our partitioning algorithm. In Section 3 we present partial strict triggering of program graph nodes. Next, a brief description of improved scheduling algorithms is given in Section 4. In Section 5 we present a model of a macro dataflow computer which supports partial strict triggering of program graph nodes. Performance evaluation of improved scheduling algorithms with execution of program graph on the macro dataflow computer sim-

ulator is done in Section 6. And, finally, Section 7 presents presents concluding remarks, as well as directions for future research work.

2 Description of the partitioning algorithm

We have used similar grain size determination algorithm as Sarkar's "internalisation" algorithm (Sarkar 1989), which clusters nodes together to minimise the schedule length on an unbounded number of processors. The algorithm initially places each task in a separate cluster and considers the arcs in descending order according to the amount of data transferred over each arc. Given arc A_{ij} connecting nodes N_i and N_j, the algorithm merges the clusters containing these nodes to 'internalise' any communications between nodes in these respective clusters. This merging step is accepted if it does not increase an estimate of the parallel execution time of the program graph on an infinite number of processors, where nodes in the same cluster are constrained to be executed on the same processor. The partitioning process in 'internalisation' algorithm is stopped when cluster's execution time is equal or greater than one percent of ideal parallel execution time. A cluster must also satisfies a convexity constraint, which ensures that a cluster can run to completion once all its inputs are available. Our opinion is that the size of clusters (execution time) is computer architecture dependent, so our partitioning algorithm stops further aggregation into bigger clusters when the size of a cluster is equal or greater than maximal granularity of cluster which depends on the organisation of the macro dataflow computer architecture.

3 Partial strict triggering of program graph nodes

Now, let us consider a group of program nodes that are connected with the precedence relation. If the sum of communication delays for transferring operands between nodes is greater than the sum of their execution times, it is possible to achieve fastest execution time with joining them in a larger program grain. Of course, there are other techniques for grain size determination, but it is interesting that joining small grains into larger one enables the processing element to start with execution of a new grain without all input operands. An example is shown in Figure 1. We aggregate fine grain program graph nodes (1, 2, 3) into a coarse grained program graph node. The new node N can start with the execution on a free processing unit immediately after operand number 1 is available, although operand number 2 is not present. This operand must be available at least 10 time units after the beginning of the node N execution to avoid the node execution delay. The operand number 4 is sent to the successor nodes 20 time units before the end of the execution of node N, so the triggering

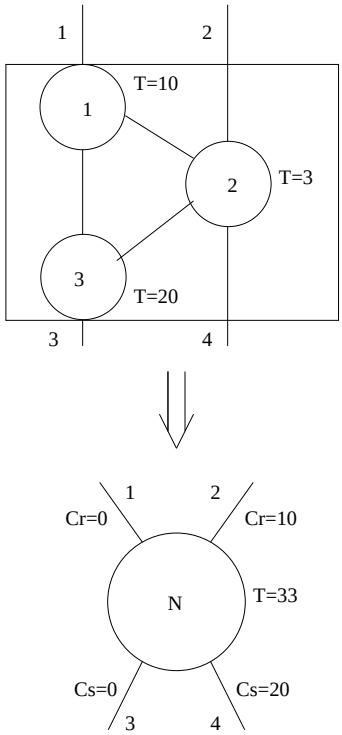


Figure 1: Example of partial strict triggered node

of successors which use this operand is faster.

We have introduced two new attributes for each communication arc. C_S represents operand's relative sending time. It defines the time from the moment when the operand is sent to the end of a node execution. C_R represents operand's relative receiving time. It defines the time from beginning of a program graph node execution to the moment when this operand must be present to avoid delaying the node's execution. Operands which have C_R equal to zero are called *strict operands* and must be present before the execution of their program graph nodes. Operands which have C_R greater than zero are called *non-strict operands* and must be present between time from beginning of a program graph node execution and C_R.

The following notation is used throughout this paper:

G	: graph $G(N, E)$,
$N(G)$	: set of nodes in G,
$E(G)$	: set of arcs in G,
n	: number of nodes in in G,
n_i	: i-th node in $G; i = 1..n$,
$T(n_i)$	: execution time of node n_i,
$N_s(n_i)$	: set of successors of the node n_i,
ns_i	: number of successors of the node n_i,
$O(G)$	: set of output nodes of graph G,
$CP(n_j)$	: the length of exit path for node n_j,
$I(n_{ij})$	: set of input operands of n_j that originates from n_i,

$C(n_i, n_j, nii, noj)$: communication delay time between nodes n_i and n_j, n_i is source and n_j is destination, nii is a number of particular input operand of node n_j, noj is a

number of particular output operand of node n_i,

$C_S(n_i, n_j, nii, noj)$: relative sending time of ope-rand between nodes n_i and n_j, n_i is source and n_j is destination, nii is a number of particular input operand of node n_j, noj is a number of particular output operand of node n_i,

$C_R(n_i, n_j, nii, noj)$: relative receiving time of operand between nodes n_i and n_j, n_i is source and n_j is destination, nii is a number of parti- cular input operand of node n_j, noj is a number of particular output operand of node n_i.

4 Description of scheduling algorithms with partial strict triggering of program graph nodes

CPM scheduling algorithm (Kohler 1975, Ojsteršek 1994, Shirazi et al. 1990) assigns the program graph nodes to the processing elements on the basis of the priority list scheduling method. A priority weight for each node is a length of the longest path from program graph node to the terminal node. The original CPM algorithm computes the priority weight (the length of exit path) of each node without using communication delay time for transferring operands between nodes. We have defined new priority weight which is computed by using execution time of a node and communication delay times C_S and C_R of its input and output operands. The length of exit path for node n_i ($CP(n_i)$) is computed in the following way:

$$\begin{aligned}&\forall n_i \in O(G) : CP(n_i) = T(n_i)\\ &\forall n_i \notin O(G) \wedge \forall n_j \in N_s(n_i) \wedge \forall n_j \; \exists \, CP(n_j) :\\ &CP(n_i) = MAX_{j=1,ns_i}(CP(n_j)\\ &\qquad +C(n_i, n_j, nii, noj) - C_R(n_i, n_j, nii, noj)\\ &\qquad -C_S(n_i, n_j, nii, noj) + T(n_i)) \qquad (4.1)\end{aligned}$$

In the improved CPM algorithm we also make sure that all nonstrict operands are present in time from beginning of a program graph node execution to the moment when these operands must be present to avoid delaying the node's execution. If we can't assure this condition for nonstrict operand, we assign its C_R to zero. Time complexity of original CPM algorithm is $O(n^2)$, where n is number of program graph nodes. Time complexity of improved CPM algorithm is also $O(n^2)$.

Vertically Layered (VL) scheduling algorithm (Hurson et al. 1990, Kvas et al. 1994, Ojsteršek 1994) is based on two philosophies:

1. assigning concurrently executable nodes to separate processing units and
2. assigning nodes connected serially to the same processing element.

Main idea of this algorithm is the distribution of program graph nodes into vertical layers, where nodes constituting a single layer, can be allocated to a processing element. Actual allocation is done in two phases: the separation and optimisation phase. In separation phase critical path (CP) is identified and nodes on CP are assigned to one vertical layer. Rearranging is done in an iterative manner as follows: nodes on longest directed path emanating from an arc in a node that is already assigned, are assigned to the next available vertical layer.

The separation phase does not take into account communication delays among processing elements (PEs). The optimisation phase rearranges nodes by considering inter-PE communication delays. For example, if two subsets of nodes are arranged in two distinct vertical layers, and there is the transitory relationship between them, there will be inter-PE communication costs associated with the execution of two vertical layers. In order to improve overall execution time, we can consider combining the subsets of nodes into a single vertical layer. This eliminates the communication time between two layers and the overall execution time is now the sum of execution times of subsets. However, if the new execution time results in a larger delay, two subsets are assigned to different PEs.

We improved VL algorithm optimisation phase that rearranges nodes by considering communication delay times C_S and C_R. Original equations (Hurson et al. 1990) that are used to compare communication and execution time between nodes contain simple communication time delays. Now, if we use partial strict triggering, we can change the communication delays in the following way:

$$\begin{aligned}C_u(n_i, n_j) = MAX_{nii \, \in \, I(n_{ij})}(&C(n_i, n_j, nii, noj)\\ &-C_S(n_i, n_j, nii, noj) - C_R(n_i, n_j, nii, noj))\end{aligned} \qquad (4.2)$$

Then we use C_u in equations instead of C. We also assure that all nonstrict operands are present in time from beginning of a program graph node execution to the moment when these operands must be present to avoid delaying the node's execution. If we can't assure this condition for nonstrict operand, we assign its C_R to zero. Time complexity of original VL algorithm is $O(n^4)$, where n is number of program graph nodes. Time complexity of improved VL algorithm is also $O(n^4)$.

Duplication scheduling heuristic algorithm (Benko et al. 1995, Kruaratrachue & Lewis 1988, Ojsteršek 1994) maximises parallelism and mini-mises communication delays by insertion of ready program graph nodes into idle time slots of Gantt chart and by duplication of critical program graph nodes. We have focused our research efforts to three main extensions of the original heuristic:

– We have converted program graphs that have nonstrict operands into program graphs that have all strict operands. A new type of program graph is called a strictly triggered program graph. Each program graph node of original program graph is transformed into several nodes of strictly triggered program graph. We call them a group of strict nodes. All nodes of a group

have to be executed on the same PE. PE pre-empts the execution of a group if in the execution phase a nonstrict operand is not present. While this operand is not present, PE executes other groups. When a nonstrict operand arrives PE continues with the execution of the pre-empted group.

– We proposed a new processing units allocation scheme, which greatly improves the efficiency of scheduling. An original DSH algorithm uses a length of longest path from the node to the exit node and number of immediate successors as a priority weight for processor allocation. We introduced predecessor selection scheme (PSS) and successor selection scheme (SSS). Both selection schemes are trying to decrease the amount of communication overhead between nodes by scheduling *similar* tasks on the same processing unit. The term *similar* is used to refer to tasks that are expected to communicate with each other, either directly or indirectly. PSS considers two tasks as similar, if one of the tasks is a direct or an indirect predecessor of the other one. SSS considers two tasks as similar if they have at least one successor task in common, i.e. they both send at least one message to the same destination.

– We have also extended DSH algorithm to take into account a finite number of communication channels between processing units. In this way, the accuracy of the results obtained from simulation of coarse grained program graph execution on the model of parallel computer has been improved.

Time complexity of original DSH algorithm is $O(n^4)$, where n is number of program graph nodes. Time complexity of improved DSH algorithm is $O(n^6)$.

5 The model of macro dataflow computer

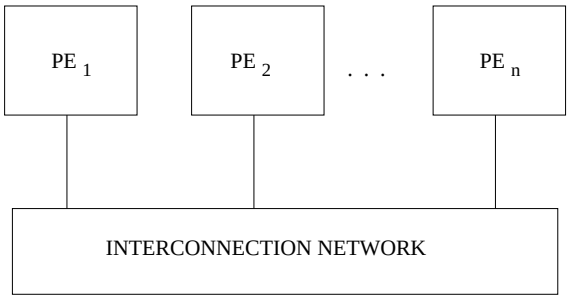


Figure 2: The Model of a Macro Dataflow Computer

We have introduced our model of a macro dataflow computer (Figure 2) which supports partial strict triggering of coarse grained program graph nodes. Our model of a macro dataflow computer (MMDC) is a loosely coupled multiprocessor with processing elements (PE) and an interconnection network.

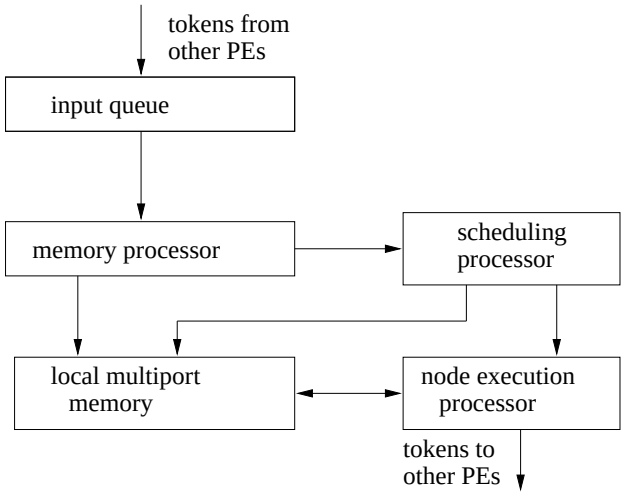


Figure 3: A Structure of a Processing Element of the MMDC

Every PE (Figure 3) contains an input queue, a local multiport memory, a memory processor, a scheduling processor and a node execution processor. Program graphs are loaded on the model of a macro dataflow computer at the beginning of their execution. Inputs of nodes (tokens) are matched together in the memory processor. When all required inputs of a particular node are present, the memory processor forms the executable node (activity) and sends a node number to the scheduling processor. The scheduling processor selects the executable nodes on the basis of a Gantt chart which has been produced by the static scheduling algorithm. It sends activities to the node execution processor. The node execution processor executes activities. If a nonstrict operand of executed activity have not arrived in time when it is required, the node execution processor preempts execution of this activity and executes another activity. When the nonstrict operand of preempted activity arrives in the memory processor, the memory processor sends the address where the activity has been preempted into queue of preempted activities, which are in multiport memory. When the node execution processor finishes with the execution of temporary executed activity, it then continues with the execution of the preempted activity. The node execution processor also sends results of computed activities to memory processor if the successor node is in the same PE or to other PEs.

In our simulation we assume that the model of a macro dataflow computer has the following features:

– All PEs have equal performances.

– A PE executes nodes and communicates with other units simultaneously.

– All PEs are connected with communication network, which has a finite number of communication channels.

– A memory space, needed for matching tokens in sets of tokens, forming activities, scheduling and executing activities of program graphs that are executed on the MMDC, must be less than the capacity of available memory space in the model of the MMDC.

6 Performance evaluation of improved scheduling

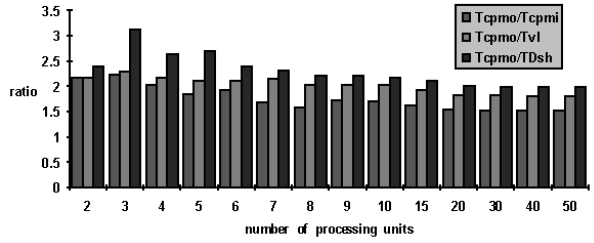


Figure 4: Comparison of improved scheduling algorithms with original CPM algorithm. The total execution time of program graph produced with usage of: (1) original CPM scheduling algorithm is represented by $T_{cpmo}(PG)$, (2) improved CPM scheduling algorithm is represented by $T_{cpmi}(PG)$, (3) improved VL scheduling algorithm is represented by $T_{vl}(PG)$, (4) improved DSH scheduling algorithm is represented by $T_{dsh}(PG)$.

We have evaluated improved scheduling algorithms with a simulation of a program graph execution on the MMDC. Program graph consisted of 200 nodes and 374 operands (306 strict operands and 68 nonstrict operands). Its level of granularity (average execution time of nodes/average communication delay time of operands) has been 0.092. We have used program graph and Gantt charts produced by original CPM algorithm and improved algorithms as simulation inputs. A comparison of ratios between total execution times of program graph with usage of original CPM algorithm and improved scheduling algorithms is depicted in Figure 4. As shown in Figure 4 all improved scheduling algorithms outperform original CPM algorithm. Improvements have been from 35% to 60%. From Figure 4 we can see that the improved DSH algorithm gave the best improvement but it have the highest computational complexity ($O(n^6)$). VL algorithm obtained better results than improved CPM algorithm, but it has higher computational complexity than the CPM algorithm.

We have observed that improvements of our algorithms are higher if level of granularity is lower than one. If level of granularity is greater than one then improvements are only few percents. In that case we use improved CPM algorithm, because it has the lowest computation complexity.

7 Conclusion

This paper presented performance evaluation of three different static scheduling algorithms improved with partial strict triggering of program graph nodes. The new model of a macro dataflow computer which supports partial strict triggering of program graph nodes is also presented. Simulation results of program graph execution on the MMDC with usage of Gantt charts produced by improved scheduling algorithms showed promising improvement over original CPM algorithm.

In our previous work we build an integrated programming environment for static partitioning and scheduling of time critical tasks which are executed on the macro dataflow realtime computer (Ojsteršek & Žumer 1992). In our future research we will incorporate scheduling algorithms into the new version of integrated programming environment.

References

[1] Benko B., Ojsteršek M., Žumer V. (1995) Improvement of Duplication Scheduling Heuristic Algorithm with Nonstrict Triggering of Program Graph Nodes. *Proceedings of First Aizu International Symposium on Parallel Algorithms/Architecture Synthesis*, Aizu-Wakamatsu, Fukushima, Japan, IEEE Computer Society Press, p. 227-233.

[2] Darbha S. and Agrawal D. P. (1996) Optimal Scheduling Algorithm for Distributed-Memory Machines. *IEEE Trans. on Parallel and Distributed Systems*, January, vol. 9, No. 1, p. 87-95.

[3] Gerasoulis A. and Yang T. (1992) A Comparison of Clustering Heuristics for Scheduling Directed Acyclic Graphs on Multiprocessors. *J. Parallel and Distributed Computing*, vol. 16, p. 276-291.

[4] Hurson A. R., Lee B., Shirazi B., Wang M. (1990) A Program Allocation Scheme for Data Flow Computers. *Proc. of the 1990 Intern. Conf. on Parallel Processing*, University Park, Peen, Pennsylvania State Univ., vol. 1, p. 415-423.

[5] Kohler W. H. (1975) A Preliminary Evaluation of the Critical Path Method for Scheduling Tasks on Multiprocessor Systems. *IEEE Trans. on Computers*, December, p. 1235-1238.

[6] Kruaratrachue B., Lewis T. (1988) Grain Size Determination for Parallel Processing. *IEEE Software*, January, p. 23-33.

[7] Kvas A., Ojsteršek M., Žumer V. (1994) Eva-luation of Static Program Allocation Schemes for Macro Dataflow Computer. *Proceedings of the 20th EUROMICRO Conference*, Liverpool, England, IEEE Computer Society Press, p. 573-580.

[8] Ojsteršek M., V. Žumer V. (1992) Improving a Time Critical Task Execution Time Using an IPRESPS. Microprocessing and Microprogramming, Amsterdam, 34, 1-5, p. 197-200.

[9] Ojsteršek M. (1994) Partitioning and Scheduling Program Graphs onto Parallel Computer System. PhD thessis. University of Maribor, Faculty of Technical Sciences Maribor, June 1994 (in slovene).

[10] Palis M. A., Liou J. C., Wei D. S. L. (1996) Task Clustering and Scheduling for Distributed Memory Parallel Architectures. *IEEE Trans. on Parallel and Distributed Systems*, January, vol. 7, No. 1, p. 46-55.

[11] Park C.-I. and Choe T.-Y. (2002) An Optimal Scheduling Algorithm Based on Task Duplication. *IEEE Trans. on Computers*, April, vol. 51, No. 4, p. 444-448.

[12] Sarkar V. (1989) Partitioning and Scheduling Parallel Programs for Execution on MultiProcessors, MIT Press, 1989.

[13] Shirazi B., Wang M., Pathak G. (1990) Analysis and Evaluation of Heuristic Methods for Static Task Scheduling. *Journal of Parallel and Distributed Computing*, October, No. 10, p. 222-232.

Dynamic Materialized View Selection in Data Warehouse Environment

Chuan Zhang
TransACT Communications, 470 Northbourne Avenue, Dickson ACT 2602, Australia
chuan.zhang@transact.com.au

Jian Yang
Infolab, Tilburg University, PO Box 90153, 5000 LE, Tilburg, Netherlands
Jian@uvt.nl

Kamalakar Karlapalem
International Institute of Information Technology, Gachibowli, Hyderbad 500019, India

Keywords: materialized views, dynamic view selection, data warehousing

Received: May 12, 2002

Materialized view selection is one of the important areas in data warehouse design. Due to the dynamic nature of data warehouse, materialized views should evolve in response to the application requirement change, which is studied as a dynamic materialized view selection problem. One aspect of dynamic materialized view selection that differs from static materialized view selection is that the existing materialized views should be taken into consideration when selecting new materialized views. In this paper, a framework is presented to illustrate the design methodology for dynamic materialized view selection. A new cost - the materialized view reorganization cost is introduced and incorporated when selecting new views to materialize. Based on our cost model, a set of algorithms for dynamic materialized view selection are developed and some experimental results are provided.

1 Introduction

1.1 Dynamic Materialized View Selection

Data warehousing is an approach to the integration of data from multiple, possibly very large, distributed, heterogeneous databases and other information sources. A Data Warehouse (DW) is a repository of integrated information available for querying and analysis. To avoid accessing the original data sources and increase the efficiency of queries processing in WD, some intermediate results in the query processing are stored in DW. These intermediate results stored in DW are called materialized views. In some ways, a DW can be seen as a set of materialized views over the data extracted from the distributed heterogeneous databases. Among many research issues related to DW design as pointed out in [11], materialized view selection is one of the most challenging problems. On one side, the materialized views speed up query processing; On the other side, the materialized views have to be refreshed when changes occur to the data sources. Therefore, two costs need to be taken into consideration in selecting materialized views: the query processing cost and the materialized view maintenance cost. In order to archive high query performance, we should store all the query results in DW. However, the view maintenance cost might be prohibitively high. The challenge is to select the materialized views that make the sum of the query processing and view maintenance cost minimal. We call it *static* materialized view selection.

Moreover, due to the dynamic nature of a data warehouse, materialized views should change in response to the changes of the application requirement. This means, as the user queries change, the static selected materialized views are very quickly outdated. Two issues arise here: first, how can we get the new materialized views without recomputing from scratch? Second, how fast can we materialize these newly selected views? In this paper, we mainly address the first issue, which is studied as a *dynamic* materialized view selection problem.

Our basic design principle is that the materialized view selection for the new set of queries should be based on the existing resources, i.e., the already materialized views whenever possible. This philosophy distinguishes dynamic materialized view selection from the static materialized view selection that has been studied most extensively.

There are reasons for not recomputing the new materialized views from scratch. First, recomputing from scratch is a waste of the existing materialized views. Second, in a frequently changing DW environment, if we choose to recompute the new materialized views which make the total cost (materialized view's query and maintenance costs) minimized, this may not necessarily the most efficient choice if the *generation cost* for these new materialized views is taken into consideration. If the query and maintenance cost

saving made by introducing these new materialized views is less than the *generation cost* of the new materialized views, the new selected materialized views are obviously not the good choice. Therefore, when we deal with the dynamic materialized view selection, we cannot ignore the generation cost of new materialized views.

Based on the above observation, the *materialized view reorganization cost* is introduced in this paper, which is the cost of generating new materialized views from the old materialized views and the base relations.

In this paper, three costs will be considered, i.e., query cost, materialized view maintenance cost, and materialized view reorganization cost. The objective of dynamic materialized view selection is to select a a set of new materialized views that makes the sum of the three costs minimal based on the assumption that the application requirements of the designed DW change relatively frequently.

1.2 Related work

There are some similar work has been done in the area of dynamic materialized view selection. A system called DynaMat in [3] unifies the view selection and the view maintenance problems under a single framework using a novel "goodness" measure for the materialized view. They distinguish operational phases of the system into two. One is the query answering phase during which a $Fragment\ Locator$ is applied to determine whether or not materialized results can be efficiently used to answer the query. The other is the update phase, during which updates received from the data sources get stored in the warehouse and materialized results in the pool get refreshed. However, they deal with the materialized view selection and materialized view maintenance separately in the two operational phrases. The work done in [10] deals with the issue of incrementally designing a DW, which selects the new views to materialize that fit in the extra space and minimizes the combined evaluation costs of the new queries and maintenance cost of the new views. With this approach they can not guarantee that the total cost of all the query (new and old) evaluation and maintenance of all the views (new and old) is minimized. Moreover, they have not considered the materialized view reorganization cost.

Our problem is different from the view adaptation problem [2]. The view adaptation tried to adapt the view in response to changes in the view definition [4, 6, 5]. The view adaptation approach recomputes the new materialization using the old materialization and the base relations. What distinguishes our problem from the view adaptation problem is as follows:

- The primary objective of adapting views is to minimize both communication and computation costs [4]. While our problem takes into consideration all three costs (query, materialized view maintenance and reorganization costs).
- View adaptation only considers the view definition change while our problem considers not only the view definition change but also other changes in the application requirements such as query frequency changes, or addition of new queries.
- The view adaptation problem usually enumerates all the views which can be adapted when view definitions are changed, i.e., all the possible redefinitions of `SQL SELECT-FROM-WHERE, GORUPBY, HAVING, UNION, and EXCEPT` views, and then see how these views can be adapted using the old materialized views. Our approach is completely based on the cost model. That is, which views should be materialized purely depends on the three costs.
- The view adaptation problem always loses the old materialized view [2] while our approach might keep some or all the old materialized views, depending on the nature of changes in the application requirement.

1.3 Contribution and organization of this paper

The contribution of our paper is of the following:

- A framework for addressing dynamic materialized view selection is developed.
- A new cost, materialized view reorganization cost, is introduced and analyzed in the context of dynamic materialized view selection.
- A set of algorithms for dynamic materialized view selection are provided and evaluated.

The rest of the paper is organized as follows: Section 2 defines and specifies the **Multiple View Processing Plan (MVPP)**, which will be used as a vehicle for cost analysis. The cost model for dynamic materialized view selection is presented in Section 3. A framework for dynamic materialized view selection is analyzed and provided in Section 4. Preprocessing and a heuristic algorithm for materialized view evolution is developed in Section 5. A genetic algorithm and its application to our problem is introduced in Section 6. Section 7 presents the experiment results and finally the conclusion and future research are given in Section 8.

2 Specification and analysis of MVPP

2.1 A motivating example

Our example is taken from a data warehouse application used in [12], which analyzes trends in sales and supply. The relations and the attributes of the schema for this application are:

```
Item(I_id, I_name, I_price)
Part(P_id, P_name, I_id)
Supplier(S_id, S_name,P_id, City, Cost, Preference)
Sales(I_id, Month, Year, Amount)
```

Suppose there are five queries, as follows:

```
Q1: Select  P_id, min(cost), max(cost)
    From Part, Supplier
    Where  Part.P_id=Supplier.P_id
          And  P_name in {"spark_plug", "gas_kit" }
          Group by P_id
Q2: Select  I_id, sum(amount*number*min_cost)
    From Item, Sales, Part
    Where  I_name in {"MAZDA","NISSEN","TOYOTA" }
          And year=1996
          And  Item.I_id=Sales.I_id
          And  Item.I_id=Part.I_id
          And  Part.P_id=
                   (Select P_id, min(cost) as min_cost
                    From Supplier
                    Group by  P_id)
          Group by  I_id
Q3:Select  P_id, month sum(amount)
   From Item, Sales, Part
   Where  I_name in {"MAZDA", "NISSEN", "TOYOTA" }
         And year=1996
         And  Item.I_id=Sales.I_id
         And  Part.I_id=Item.I_id
         Group by   P_id , month
Q4: Select  I_id, Sum(amount *I_price)
    From Item, Sales
    Where  I_name in  {"MAZDA", "NISSEN", "TOYOTA"}
          And year=1996
          And  Item.I_id=Sales.I_id
          Group by   I_id
Q5: Select  I_id, avg(amount*I_price)
    From Item, Sales
    Where  I_name in {"MAZDA", "NISSEN", "TOYOTA"}
         and year=1996
         and Item.I_id=Sales.I_id
     Group by I_id.
```

Figure 1 represents a possible global query access plan for the five queries listed above, in which the local access plan for individual queries are merged based on the shared operations on common data sets. It is called `Multiple View Processing Plan (MVPP)` in [12].

The query access frequencies are labeled on the top of each query node. For simplicity, we assume that all the base relations `Item`, `Sales`, `Part` and `Supperlier` are updated only once for a certain period of time. In the annotations in Figure 1, we abbreviate thousand as "k", million as "m", and billion as "b". The cost for each operation node in Figure 1 is labeled at the right side of the node in terms of the number of tuples accessed. For example, the cost for obtaining `tmp3` by using `tmp1` is 36m. The number at the left side of the node indicates the size of the node. For the details of how to get the numbers in this figure, please see [12].

Now we have to decide which node(s) to be materialized so that the query cost and view maintenance cost is minimal. It is obvious from this graph that we have several alternatives for choosing the set of materialized views: e.g., (1) materialize all the application queries; (2) materialize some of the intermediate nodes (e.g., `tmp1, tmp3, tmp7 etc.`); (3) leave all the non-leaf nodes virtual. The cost for each alternative shall be calculated in terms of query processing and view maintenance.

Before getting into the details of MVPP analysis, we will first review the notation used in [12] and explain the cost model.

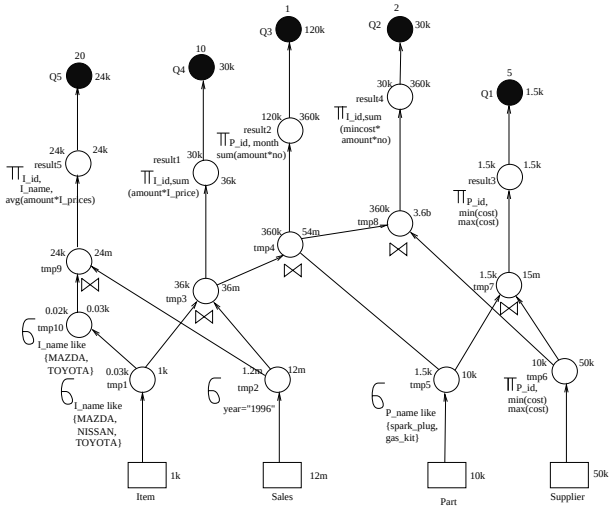


Figure 1: A motivating example

2.2 Specifications

An MVPP is a labeled DAG$(V, A, C_a^q(v), C_m^r(v), f_q, f_u)$ where V is a set of vertices which denote relational algebra operation, base relation or distinct query, and A is a set of arcs over V, such that:

- For $v \in V$, $T(v)$ is the relation generated by the corresponding vertex v.
- Let L be the set of leaf nodes.
- Let R be the set of root nodes.
- For every vertex v, let $S(v)$ denote the immediate source nodes (ancestors) that have edges pointing to v. For any $v \in L$, $S(v) = \emptyset$. Let $S^*\{v\}$ be the set of all ancestors of v.
- For every vertex v, let $D(v)$ denote the immediate successor nodes (descendants) pointed to by arcs originating at v. For any $v \in R$, $D(v) = \emptyset$. Let $D^*(v)$ be the set of all descendants of v.
- For $v \in V$, $C_a^q(v)$ is the cost incurred by query q to access $T(v)$. $C_m^r(v)$ is the cost of maintaining T(v) based on changes to the base relation $S^*(v) \bigcap L$, if $T(v)$ is materialized.
- f_q, f_u are the frequencies of query execution and view maintenance because of base relation updates respectively.

For simplicity in notation we denote the relation $T(v)$ corresponding to a vertex v as just v for the rest of the paper.

MVPP Generation: find all pairs of distinct vertices $u, v \in V$, such that $S^*(u) = S^*(v)$ and $T(u) = T(v)$, then $T(u)$ and $T(v)$ are common subexpressions which can be merged to form a MVPP.

3 The Cost Model

3.1 General Cost Analysis

Here we assume that recomputing is used whenever an update of involved base relation occurs. The following are all the necessary notation used in our cost model and algorithms.

- O_v denotes the set of global queries that use v. $O_v = R \cap D^*(v)$.
- I_v denotes the base relations that are used to produce v. $I_v = L \cap S^*(v)$.
- M denotes the existing materialized views.
- $w(v)$ denotes the static weight of a node. This is the saving achieved if node v is materialized. $w(v) = \sum_{q \in O_v}\{f_q(q) * (C_a^q(v))\} - \sum_{r \in I_v}\{f_u(r)C_m^r(v)\}$. This static weight was used in [12] to order the nodes in an MVPP for materialized view selection. The higher a node's weight, the more likely that node's selection to be materialized. However, after some nodes have been selected for materialization, using the static weight to calculate the saving is no longer valid.
- O'_v denotes the global queries that use v, but excluding those able to use any of the descendants of v that are already materialized. $O'_v = O_v - O_u$, where $u \in M \cap D^*(v)$.
- $C_s(v) = \sum_{q \in O'_v}\{f_q(q) * (C_a^q(v) - \sum_{u \in S_v \cap M} C_a^q(u))\} - \sum_{r \in I_v}\{f_u(r)C_m^r(v)\}$ denotes the dynamic weight of a node, taking into account of any existing materialized views of which v can use. The first part of C_s is the *extra* saving over and above that achieved by the existing materialized views, if v is materialized. The second part is the view maintenance cost for a materialized v. The expression $\sum_{u \in S_v \cap M} C_a^q(u)$ accounts for the saving already obtained in the case where some of the ancestors of v have already been chosen for materialization.

3.2 Materialized view reorganization

Materialized view reorganization is to generate the new materialized views from the old materialized views and the base relations. Materialized view reorganization cost basically is the processing cost of the new materialized views computed from the existing materialized views and the base relations.

Based on the above observation, we can treat the new materialized view as a query, and define the new materialized view reorganization cost as follows:

Let M' denotes the set of new materialized views. The new materialized view set might overlap with the existing materialized view set M.

The reorganization cost of a new materialized view v is defined as C_{re}^v, the query cost of v accessing $\{M \cup L\}$. M denotes the existing materialized views and L denotes the base relations.

Reorganization cost can be calculated in many ways. In this paper, we mainly consider using the following two ways:

- Calculate the reorganization cost directly as the query cost. That is, for each new materialized view v_i, the reorganization cost is the query cost of v_i accessing $\{M \cup L\}$.
- Treat it as a multiple query processing optimization problem. In this way, every new materialized view is treated as a query. Calculating the reorganization cost is similar to finding the optimal MVPP for which total query cost is minimized. We can use heuristic concatenation of optimal plans for each v_i, or formulate it as integer programming problem [12] or genetic algorithm [15].

Now the problem for dynamic materialized view selection can be described as:

Given a set of existing materialized views M, a set of base tables with new update frequencies, a set of new queries with new query frequencies, determine a set of new materialized views M' such that the sum cost of processing all the queries, maintaining all the views and reorganizing materialized view (from M to M') is the smallest possible.

4 A framework of dynamic materialized view selection

4.1 MVPP analysis

When the query application requirement changes, there are two possible scenarios for the given MVPP.

- MVPP structure does not changed.
- MVPP structure changes.

Usually the changes of the application requirements include the following situations: adding new queries, changing query frequencies, and changing query definitions. When these changes occur, they might affect the existing MVPP structure. The affection of the application requirement changes on the existing MVPP can be classified as follows:

- When new queries are added or old queries are deleted, the existing MVPP structure will change.
- When the query frequency or the updating frequency changes, the existing MVPP structure may not change. For example, in figure 1, the frequency for query $Q4$ may change from 10 to 20, the MVPP structure does not change.

– When the existing query definition changes, the existing MVPP structure may/may not change. For example, the selection criteria “year=1996" for $Q2, Q3, Q4, Q5$ may change to “year=1990", therefore, the sizes of the nodes {tmp3, tmp4, tmp8, result1, result2, result4} will changed. But the MVPP structure does not change.

  If we add one more selection criteria "I_price $>$ 20000" for $Q2$, the MVPP structure will change.

Even if the existing MVPP structure is not affected by the change, we cannot say that the existing MVPP is the optimal or near optimal from which the new materialized views are chosen under the new environment. There might be other MVPPs which are better than the existing MVPP. The question is how to evolve the existing MVPP to other better MVPPs in order to select the new materialized views. In the next subsection, we will introduce our design methodology for the dynamic materialized view selection.

4.2 The dynamic materialized view design methodology

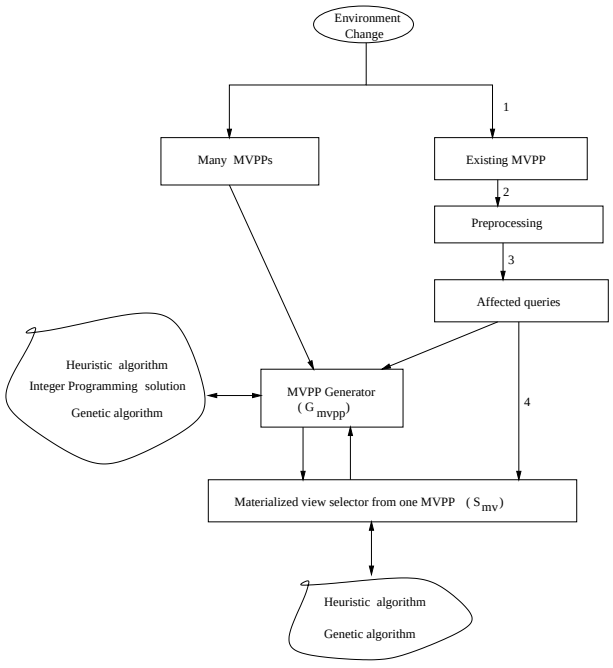


Figure 2: Framework of dynamic materialized view selection

Figure 2 is a framework of the dynamic materialized view selection. As the affection of the application requirement change to the existing MVPP is hard to measure, the existing MVPP might evolve to other better MVPPs with small changes. Thus our policy shown in figure 2 is that no matter how big or small the change is, it all depends on the warehouse administrator to decide whether to evolve the existing MVPP. Therefore there are two decision branches to begin with in Figure 2: selecting materialized views from the existing MVPP or other MVPPs.

In Figure 2, the MVPP generator (G_{mvpp}) is a tool pool which includes various algorithms to generate MVPPs from queries. These algorithms can be heuristic algorithm, 0-1 integer programming solution, or genetic algorithm, etc.

The materialized view selector (S_{mv}) is also a tool pool which includes various algorithms to select materialized views from one MVPP. These algorithms can be heuristic algorithm which will be introduced in the next section, or genetic algorithm [13].

Usually we use different combinations of the algorithms from G_{mvpp} and S_{mv} under different constraints such as computation time. For example, we can use genetic algorithm to produce the optimal MVPP and based on the generated MVPP a heuristic algorithm from S_{mv} can be applied to select materialized views. The G_{mvpp} and S_{mv} usually run interactively.

If new materialized views selection is based on the existing MVPP, preprocessing can be applied to determine which queries will be affected by adding the new queries. This preprocessing will be introduced in the next section. In this way we can re-calibrate the original problem as selecting materialized views from the affected queries instead of all the queries. The problem space shall be restricted after preprocessing.

If new materialized views selection is not based on the existing MVPP, other MVPPs should be generated and the new materialized views are selected from the newly generated MVPPs.

In the next two section, we will introduce algorithms to dynamically select materialized views from one MVPP or many MVPPs when application requirement changes occur. The results of these algorithms will be compared in the experiments in Section 7.

5 Heuristic algorithm for Dynamic Materialized View Selection

In this section, we shall firstly introduce three rules which determine the nodes in an MVPP which are not affected by the changed query set. Secondly, we present an algorithm which locate the change affecting scope within an MVPP. Thirdly, we explain the process for materialized view reorganization. The above three steps are all part of the preprocessing for dynamic materialized view selection. Finally, a heuristic algorithm to dynamic select materialized views is proposed.

5.1 The preprocessing for new materialized views selection

The following three rules can be used to determine how nodes are affected by each other in the process of material-

ization. The proof of three rules can be refereed to [14].

Rule 1: when v_1 is a descendant of v_2, and the static weight of v_1 is greater than that of v_2, if v_1 cannot be materialized, then v_2 will not be materialized.

Rule 2: if v_1 is a descendant of v_2, and the static weight of v_1 is greater than that of v_2, v_1 and v_2 are supporting the same queries, and v_1 is materialized, then there is no gain to materialize v_2.

Rule 3: If v_1 is an ancestor of v_2, and the static weight of v_1 is greater than that of v_2, v_1 and v_2 are supporting the same queries, if v_1 is materialized, then v_2 shall not be materialized.

When change occurs, there will be queries which are *directly* affected if they are overlapping with the changed queries in terms of the intermediate views. We need to further identify those queries which are not overlapping with the changed queries, but are overlapping with the existing *affected* queries. Therefore they are affected *indirectly* through the overlapping with the affected query. The algorithm in Figure 3 is devised to find the scope of the affected queries. The unaffected queries will not be considered when dynamically selecting the new materialized views.

Step 2 in Figure 3 is the process to determine the overlapping part between the changed queries and the existing queries. Any existing query has overlapping with the changed query will be affected and need to be considered for evolution. For those existing queries which have no overlapping with changed query, we apply step 3 to check if they have overlapping with affected queries and how they are overlapped.

After the above process, we get a scope of queries which are affected by the application requirement change. Within this scope, the following steps can be applied to further reduce the search space:

1. In this scope, calculate the static weight $w(v)$ of each node v except the base table.
2. For each node which weight $w(v)$ is greater than 0, calculate its reorganization cost except the base table and the existing materialized views.
3. Subtract the reorganization cost C^v_{re} from the static weight $w(v)$.
4. if $C^v_{re} - w(v)$ is greater than 0, then v will be a candidate for materialization.

The idea behind this process is that under the new environment, if a node's estimated saving (the static weight) is less than its reorganization cost, it will not be considered to be materialized. If there is no candidate for materialization, then there is no need to re-select the materialized views from this MVPP. Otherwise, based on these candidate nodes, apply algorithm in Figure 4 (which will be discussed in the next section) to re-select the materialized views.

```
begin
  1. set up two sets: Q_affected and Q_unaffected.
     Initially let Q_affected = ∅, Q_unaffected = {all the
     existing queries}
  2. for every q_i ∈ Q_unaffected
       if q_i has overlapping with the changed query
         remove q_i from Q_unaffected
         insert q_i into Q_affected
         create O_{q_i} ={overlapping parts}
       endif;
     endfor;
  3. for every q_k ∈ Q_affected
       if it has overlapping with any
       q_j ∈ Q_unaffected
         and the overlapping has common parent
         with any element in O_{q_k}
         this common parent belong to q_k and
         its weight is greater than 0
       then remove q_j from Q_unaffected
         insert q_j into Q_affected
         insert the overlapping into O_{q_k} and O_{q_j}
       endif;
     endfor;
endbegin;
```

Figure 3: A_{IQS}- Algorithm to find out the affected queries when changes happen

With this process, we can further reduce the number of nodes to be selected to materialize in an MVPP.

5.2 Heuristics for evolving materialized views in a given MVPP

Given an MVPP with some changes such as the new frequencies, the new node sizes, and a new added query, and a set of existing materialized views, we need to select a new set of materialized views so that the total cost of query processing, view maintenance, and reorganization cost is minimal based on the assumption that application requirements change quite frequently for the data warehouse. Within a particular MVPP finding the optimal set of materialized views can be done by enumerating all possible combinations of materialized views and determining which combination results in the minimum query, maintenance, and reorganization cost. However, finding the optimal set of materialized views in this manner has a complexity of O(2^n), where n is the number of views in the MVPP. Since the problem complexity is exponential, heuristics have generally been used to handle this kind of problem. In this section, we devise a heuristic to select new materialized views for the changing environment.

The dynamic materialized view selection algorithm in Figure 4 is based on the following idea: whenever a node is considered to be materialized, (note that the existing materialized view might be considered again), we calculate

```
begin
  1. M' := ∅;
  2. create list LV for all the nodes (with positive
     value of weight subtracted the
     reorganization cost based on the descending
     order of their values;
  3. pick up the first one v from LV;
  4. generate O_v, I_v, and S_v;
  5. calculate C_s - C_re^v
  6. if (C_s - C_re^v) > 0, then
       6.1. insert v into M';
       6.2. remove v from LV;
  7. else remove v and all the nodes
     listed after v from LV who are in the subtree
     rooted at v;
  8. re-calculate the weight of nodes in LV
     Remove all nodes with weight less than 0.
  9. repeat step 3 until LV = ∅;
  10. for each v ∈ M', if D(v) ⊂ M',
     then remove v from M';
end;
```

Figure 4: A Heuristic Algorithm for Materialized View Design

the dynamic weight of the node which is the actual saving it can bring if this node is materialized, compare it with the reorganization cost, if this value is positive, then this node will be materialized and added into the new materialized view set M'.

As we know, heuristic methods have well known drawbacks. Although they undoubtedly greatly reduce search effort they always run into the risk of failing to find the optimal, or perhaps even a near-optimal solution. In order to improve the result, we also develop a genetic algorithm to solve this problem, which will be illustrated in the next section.

6 Genetic algorithm for dynamic materialized view selection

6.1 Genetic algorithm (GA)

One of differences between genetic algorithms and other heuristic techniques is that genetic algorithms operate on a *population* of *individuals*, not on a single *individual*. Here an individual means a *solution*. From the prospective of materialized view selection, an individual represents the set of selected materialized views; from the prospective of MVPP generation, an individual is one MVPP. Every *population* is called a *generation*. A single solution is called a *Phenotype* represented by a single individual. Solutions are represented as *chromosomes*, which is composed of *genes* that can take different *values (alleles)*. A genetic algorithm creates an initial generation, G(0), and for each generation, G(t), generates a new one, G(t+1). An abstract view of the algorithm is shown in Figure 5.

```
begin
  Generate initial population, G(0);
  Evaluate G(0);
  t:=0;
  repeat
        t:=t+1;
        generate G(t) from G(t-1) by applying
        crossover or mutation;
        evaluate G(t) based on fitness function;
        alter G(t);
  until a solution is found;
end;
```

Figure 5: An abstract view of Genetic Algorithm

Each problem should have its own solutions represented as chromosomes by an appropriate encoding. *Selection, crossover, and mutation*, are applied to successive *populations* to create new populations, while selection is based on the value of the *fitness* function for the population. In other words these three operators are applied to G(t-1) to generate G(t) as shown in Figure 5. In order to get the best solution, quite a lot generations may be evolved. Several stopping criteria exist for the algorithm. For example, the algorithm may be halted when all solutions in a generation are identical. Or the algorithm would halt after a fixed number of evaluations and take the best solution found so far.

6.2 The representations in GA

As introduced in Section 4, the *individuals* in the two genetic algorithms G_{mvpp} and S_{mv} represent MVPP and materialized views respectively.

Given n queries $Q_1, Q_2, ...Q_n$, the fixed length of each individual is n, where n is the number of queries. An individual in G_{mvpp} is an MVPP, which can be represented as $P_{1i}, P_{2j}, ..., P_{kl}, ..., P_{mn}$, where P_{kl} indicates the kth processing plan for query Q_l.

With the set of possible processing plans for a query, we order the processing plans according to certain criteria such as ascending order of the query cost, and then obtain an array of integer numbers corresponding to the processing plans.

The representation of S_{mv} genetic algorithm based on the result - the optimal MVPP, obtained from G_{mvpp} is a DAG(Directed Acyclic Graph) rather than a binary string. If we can map the representation from a DAG to a binary string, we can apply GA to the problem of materialized view selection. The mapping strategy is depicted in Figure 6.

For example, search through the DAG in Figure 1 using width-first, we obtain the mapping array as follows {Q5, Q4, Q3, Q2, Q1, result5, result1, result2, result4, result3, tmp9, tmp3, tmp4, tmp8, tmp7, tmp10, tmp1, tmp2, tmp5, tmp6}, its length is 20 excluding the 4 source tables { Item,

begin
1. Input a MVPP represented by a DAG.
2. Use a certain graph search strategy such as breadth-first, depth-first or problem-oriented searching method, to search through all of the nodes in the DAG and produce an ordered sequence of these nodes.
3. Based on this sequence of nodes, create a array to store the sequence of 0s and 1s. 0 denotes that the corresponding node in the array is not materialized. 1 represents the corresponding node in the mapping array is materialized. This array is called the mapping array.

end;

Figure 6: A mapping strategy

Sales, Part, Supplier}.

The string {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0} means initially all the corresponding nodes are not materialized.

Suppose the result produced by a GA is {0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1}. This means the nodes Q4, Q1, result5, tmp2, tmp5, tmp6 shall be materialized. The rest nodes are not materialized.

6.3 Mapping cost function to fitness function

Since the objective in our cost model is stated as the minimization of sum of query, maintenance and reorganization cost while the fitness function of a GA is normally stated as maximization, there should be a transformation from our cost function to the fitness function. We use a commonly used transformation in genetic algorithms:

$$f(x) = \begin{cases} C_{max} \text{ - c(x)} & \text{when } c(x) < C_{max} \\ 0 & \text{otherwise} \end{cases}$$

where $c(x)$ denotes the cost function and f(x) is the fitness function. There are a lot of ways to choose the coefficient C_{max}. C_{max} may be taken as an input coefficient, as the largest $c(x)$ value observed so far, as the largest $c(x)$ value in the current population, or the largest of the last k generations.

With this transformation, we can get that the less the cost function is, the more the fitness function is.

An individual (MVPP) in G_{mvpp} is evaluated as follows:

Mapping in the S_{mv} genetic algorithm is direct. The cost is the total cost (query, maintenance, and reorganization) of a given global plan.

Discussion on operators *crossover*, *mutation*, and *selection* can be found in [14, 15].

6.4 The discussion of the algorithms

The G_{mvpp} algorithm is used to generate MVPPs from local processing plans based on queries. The G_{mvpp} algorithm can be either genetic algorithm or heuristic. Based on each produced global processing plan, the S_{mv} algorithm which is either genetic algorithm or heuristic is used to select the materialized views with minimal total cost (query, maintenance, and reorganization). The G_{mvpp} and S_{mv} should run interactively. If the G_{mvpp} algorithm is a genetic algorithm, the cost value in the S_{mv} is used as a fitness value in the G_{mvpp} genetic algorithm. In other words, for every population, G_{mvpp} will call S_{mv} to get the value for its fitness function. If the G_{mvpp} algorithm is heuristic, then the S_{mv} algorithm can take the result from G_{mvpp} and work on the generated "best" MVPP to select the views to materialize to get the minimal cost.

7 Experiment

All experiments were performed under SUNOS 5.5. The simulation software was built on the basis of the Simple Genetic Algorithm [8] and GAlib [7]. Based on the Simple Genetic Algorithm program [8] which is a C-language translation and extension of the original Pascal SGA code presented in [1], we developed a S_{mv} genetic algorithm. Since GAlib [7] is a library about genetic algorithms, it includes almost everything about representation and genetic operators, we developed G_{mvpp} genetic algorithm based on GAlib.

For simplicity, query processing plans used in our experiments are generated as a set of left-deep binary trees. It has been argued that good solutions are likely to exist among these trees [9]. The experiments were run over randomly generated queries. These queries share at least two relations.

In order to test the behavior of our algorithms, we will compare the following three approaches:

1. Apply genetic algorithm to all the queries to generate the MVPPs and based on each MVPP apply the heuristic algorithm (GA(all)-H); GA(all)-H approach looks like a recomputing from scratch. However, in this approach, the initial individuals were generated by perturbation of existing solution (MVPP), thus this approach is not recomputing from scratch. It makes use of the existing MVPP.

2. Based on the existing MVPP, after preprocessing, apply genetic algorithm on the affected queries to generate many MVPPs and based on each MVPP apply the heuristic algorithm (Pre-GA(affected)-H);

3. Based on the existing MVPP, after preprocessing, apply the heuristic algorithm in figure 4 (Pre-H);

The comparison results among the above three approaches are shown in figure 7. The costs are normalized

using the Pre-H as the reference. The results shown have been averaged over ten independent runs with different application requirement changes at 10, 15, 20, 25, 30 queries. The application requirement changes considered here are query frequency change and adding new queries. All these changes are randomly generated.

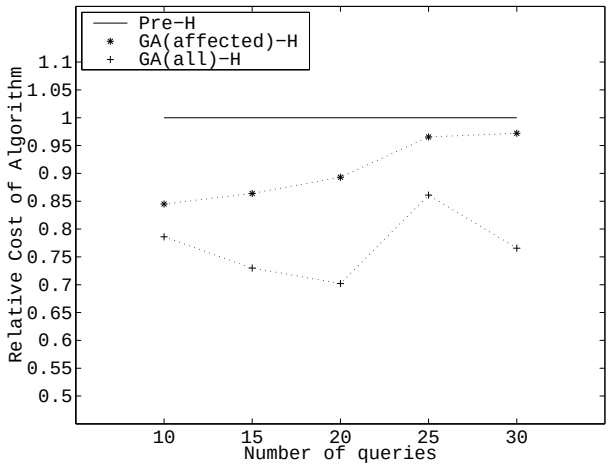


Figure 7: The comparison of three approaches

From the Figure 7, we can observe that the GA(all)-H approach is the best among the three approaches in terms of the total cost discussed before, i.e., the sum of the query access costs, the maintenance costs, and view reorganization costs. This is not surprising because GA(all)-H approach generate many MVPPs, the other two approaches just select materialized views from the existing MVPP. Although GA(all)-H is the best of the three approaches, however it consumes the longest time.

It can also be observed from the experiment process that with the materialized view reorganization cost, all three approaches can converge to the final solution faster than those recomputing approach. It can be inferred that with the materialized view reorganization cost, the solution search space is reduced.

8 Conclusion

The dynamic materialized view selection based on multiple queries is a hard combinatorial optimization problem. In this paper, a framework for dynamic materialized view selection is proposed. In order to avoid recomputing from the scratch, a reorganization cost is introduced as a measurement for materialized view evolution. Based on the proposed cost model, a set of algorithms are devised to evolve the existing materialized views. The experimental results show that the introduction of a genetic algorithm to our problem can reduce the total cost significantly. Our study also shows that with some small application requirement changes the existing MVPP might evolve to other MVPPs.

In [13], we have studied how application requirement change can affect the existing materialized views in the existing MVPP. After this study, we conclude that the way that the existing MVPP is affected by the application requirement change is unpredictable and hard to measure. Therefore we developed a set of measurements or rules to assess how the application requirement changes affect the existing MVPP and which queries need to be considered for re-generating MVPPs and selecting materialized nodes based on the generated MVPP. We believe our study provides a foundation for future work on dynamic materialized view selection.

References

[1] D.E. Goldberg. Genetic algorithms in search, optimization and machine learning. *Addison Wesley*, Reading(MA), 1989.

[2] Ashid Gupta, Inderpal Singh Mumick, and Kenneth A. Ross. Adapting materialized views after redefinitions: Techniques and a performance study. *Proc. of the ACM SIGMOD international Conference of Management of Data*, pages 211–222, 1995.

[3] Yannis Kotidis and Nick Roussopoulos. Dynamat: A dynamic view management system for data warehouses. *Proc. of the ACM SIGMOD international Conference of Management of Data*, pages 371–382, 1999.

[4] M. K. Mohania. Avoiding re-computation: View adaptation in data warehouses. *8th International Database Workshop, Hong Kong, Springer-Verlag*, pages 151–165, 1997.

[5] Mukesh K. Mohania and Guozhu Dong. Algorithms for adapting materialized views in data warehouses. *International Symposium on Cooperative Database Systems for Advanced Applications, Kyoto, Japan*, pages 62–69, 1996.

[6] Mukesh K. Mohania and Guozhu Dong. Materialized view adaption in distributed databases. *Asian Computing Science Conference (ASIAN)*, pages 353–354, 1996.

[7] Massachusetts Institute of Technology. MIT. Galib: A c++ genetic algorithms library. *http://lancet.mit.edu/galib-2.4/GAlib.html*, v2.4, 1998.

[8] Robert E Smith, David E. Goldberg, and Jeff A. Earickson. SGA-C: A C-language implementation of simple genetic algorithm. *TCGA Report No. 91002*, March,1994.

[9] Michael Steinbrunn, Guido Moerkotte, and Alfons Kemper. Heuristic and randomized optimization for the join ordering problem. *VLDB Journal*, 6(3):191–208, 1997.

[10] Dimitri Theodorators and Timos K. Sellis. Dynamic data warehouse design. *DaWak'99*, pages 126–135, 1999.

[11] Jennifer Widom. Research problems in data warehouse. *Proceedings of 4th International Conference on Information and Knowledge Management*, pages 25–30, 1995.

[12] Jian Yang, Kamalakar Karlapalem, and Qing Li. Algorithm for materialized view design in data warehousing environment. *Proceedings of 23th International Conference on Very Large Data Bases (VLDB)*, pages 136–145, 1997.

[13] Chuan Zhang and Jian Yang. Genetic algorithm for materialized view selection in data warehouse environments. *1st International Conference on Data Warehousing and Knowledge Discovery(DaWak'99)*, Lecture Notes in Computer Science, Italy, 1999.

[14] Chuan Zhang and Jian Yang. Materialized view evolution support in data warehouse environment. *Database Systems for Advanced Applications (DASFAA)*, Taiwan:247–254, 1999.

[15] Chuan Zhang, Xin Yao, and Jian Yang. An evolutionary approach to materialized views selection in a data warehouse environment. *IEEE Trans on Systems, Man, and Cybernetics (Part C)*, 31(3):P295–303, 2001.

Engineering Software by Grammatical Inference

Erkki Mäkinen
Department of Computer and Information Sciences
FIN-33014 University of Tampere, Finland
em@cs.uta.fi

Tarja Systä
Software Systems Laboratory, Tampere University of Technology
P.O. Box 553, FIN-33101 Tampere, Finland
tsysta@cs.tut.fi

Keywords: software engineering, grammatical inference, design

Received: June 7, 2003

Practical applicability of grammatical inference is considered by studying its use for various tasks and modeling problems in software engineering. It is found out that interactive inference methods are potential for "design-by-example" tasks. The required user involvement can be used to avoid undesired results. It also helps the user to recognize the key problems in the application domain. Automatic inference algorithms based on identification in the limit, in turn, are suitable for "comprehension tasks" like reverse engineering existing software systems, where the constructed model represents an overview of the systems to be studied rather than an explicit design to be aimed at.

1 Introduction

Inductive inference denotes the process of hypothesizing a general rule from examples. In grammatical inference, the inference process is related to (formal) languages and to their generating and accepting devices. Most of the early research (see for example [1]) on grammatical inference was motivated by problems concerning children's ability to learn natural language and its modeling in artificial intelligence. Later, grammatical inference has found applications, among other fields, in speech recognition [2], pattern recognition [3], and biological computing [4]. The purpose of this paper is to consider the applicability of grammatical inference in various design and modeling problems in software engineering. We use the word "design" here in a broader sense than usually understood in software engineering. When referring to the design phase in software development process we use the phrase "software design". Our special interest is in the behavioral aspects of software engineering.

Rumbaugh et al. commented the process of inferring general models from examples as follows: "Even a large collection of examples, however, necessarily falls short of a definitive description. [...] It is logically impossible to induce the general case from a set of examples, but well-chosen prototypes are the way most people think [5], pp. 16–17." Although the general case is unreachable also in terms of grammatical inference in a certain technical sense, we can support the user in his/her task of transforming examples into models. There exists some literature on applying grammatical inference in different software design and analysis tasks (see e.g. [6, 7, 8, 9, 10]), but, as far as we know, all the existing literature is closely related to specific applications, and general discussion concerning the use of grammatical inference as a design and modeling tool seems to be missing. A typical goal of a software design process is to define a finite automaton (or some other entity obeying a closely related formalism) modeling the behavior of the target concept. The behavioral modeling of a software design process often proceeds by first listing "normal" cases of the desired functioning as example scenarios. The goal is sharpened by given "special cases", which should also be possible in the resulting model. Based on these sample scenarios, the designer then constructs the model, typically a finite automaton. By forgetting the technical details for a while, we notice how this "design-by-example" process closely resembles that of grammatical inference: automata are inferred from samples such that the resulting devices are consistent with the given samples. Correspondingly, in dynamic reverse engineering, functionality and services of (typically Web-based) components can be concluded from their example usage [11]. More generally, the overall behavior of a software component can be learned from examples of interactions (not only concentrating on user actions) [12]. Hence, in this paper we consider grammatical inference in two somewhat different tasks: (1) as a design method used in "design-by-example" processes, and (2) as a "comprehension tool" in problems like software reverse engineering or process recovery (to be discussed later) where the overall behavior of an on-going system is to be inferred from its sample executions. It turns out that different methods of grammatical inference are applicable

in these different tasks. This paper is organized as follows. In Section 2, we shortly recall different approaches of grammatical inference. In Section 3, we introduce existing applications of inductive inference in software engineering. Section 4 is devoted to a general discussion on the use of grammatical inference as a design and modeling method. Finally in Section 5 we make some concluding remarks and give suggestions for further work.

2 Grammatical inference

In this section, we shortly recall the basic approaches of grammatical inference. We use the terminology of *grammatical* inference since a finite automaton is a natural form for the result of a design process. Moreover, most existing applications in software engineering obey the grammatical view of inference. We restrict ourselves to exact inference methods and neglect approximate methods, since the algorithms related to the exact inference methods are simpler. Identification in the limit [1] views grammatical inference as an infinite process. The inference algorithm is supposed to obtain an infinite sequence of input strings, and after reading each string, the algorithm outputs a hypothesis. The inference process is successful if the hypotheses "converge" to the correct language, that is, if the hypotheses eventually keep unchanged and they define the correct language. The input strings can contain only words of the desired language (inference from positive data) or all strings over the alphabet in question with an additional label indicating whether they are in the desired language (inference from positive and negative data). The infinite nature of identification in the limit seems to imply that it is not a suitable model for a practical design algorithm. Moreover, Gold's [1] famous result says that regular languages are not inferable in the limit from positive data only. However, identicifation in the limit is successfully used in "comprehension tools". In the next section we also briefly introduce a design application where identification in the limit is strengthen by an exhaustive search method. Identification in the limit typically assumes that samples are arbitrary or random [13]. From the point of view of general design and modeling tasks, this assumption is not valid since instead of arbitrary samples the desinger can choose as descriptive samples as he/she wants. Hence, it is reasonable to investigate inference approaches that assume the set of input samples to be "helpful". Angluin's model of Minimally Adequate Teacher (MAT) [14] is such an approach. MAT consists of an oracle, called the Teacher, who is capable of answering two kinds of questions posed by the Learner. The purpose of *membership queries* is to increase the Learner's knowledge of the language to be inferred. Technically this means that the Learner fills up a so called *observation table*, which contains the present information concerning members and non-members of the desired language. When the observation table fulfils a certain condition, a finite automaton is constructed as a hypothesis. An *equivalence query* tests whether the language defined by the hypothesis is equal to the desired language, and provides a *counterexample* if not. The counterexample is a word w from the symmetric difference of the desired language L and the language K defined by the hypothesis. Hence, w belongs either to L or to K, but not to both. When using MAT as a design tool, the (human) designer plays the role of the Teacher by answering membership and equivalence queries posed by the algorithm which plays the role of the Learner. MAT is capable of inferring all regular languages [14]. Efficient implementations of MAT are discussed by Balcázar *et al.* [15] and Rivest and Schapire [16]. It is instructive to notice the difference between our approach (grammatical inference as a general design and modeling tool) and *teaching* in the sense of Goldman and Mathias [17] and Mathias [18]. In teaching, the Teacher completely knows the concept to be learnt by the Learner and the task is to "transfer" as efficiently as possible the information to the Learner. In our case, the Teacher (the designer) does not completely know the desired concept. If the Learner knew, he/she would not need the design tool. Instead, the Learner knows certain separate use cases from which the algorithm infers the desired concept in the form of a finite automaton. The main problem of modeling the teaching situation is *collusion* [17, 18], where the Teacher tells "too much" to the Learner by simply encoding the target concept. This problem is not relevant in our case, since we are not interested in modeling the situation *per se*, but in constructing the desired automaton.

User's view of the desired concept may change, or at least may become clearer, during the process with the help of interaction between the designer and the design tool. In addition, the user typically fine tunes or makes possibly major additions to the concept later on. The fact that he/she does not know the desired concept is the source of various downsides: for instance, the user does not know when the resulting concept is complete, that is, when the set of example scenarios is covering enough. This can be problematic also when the method is applied to software design or reverse engineering.

3 Examples of applying inductive inference in software engineering

In this section we briefly introduce five different existing applications of inductive inference in the area of software engineering. We start with an application, an induction theory of software testing, which does not use the grammatical but functional inference terminology. Zhu [19, 20] sees the nature of software testing as inductive inference where adequacy criteria are interpreted as convergence criteria of inductive inference. By doing so, he results in new expressions of the relationship between software testing and software quality. Another inference application is that of *process recovery*, which resembles reverse engineering. Developing a formal model for an on-going, complex pro-

cess can be difficult, costly, and error prone. Cook [6] and Cook and Wolf [7] have developed a data analysis technique called process discovery for solving this problem. The name of the method emphasizes its main principle of finding processes and modeling their behavior. The methods used also require human guidance in the form of tuning the parameters, and selecting and applying of the event data. Cook and Wolf [7] argue that in addition to software processes, their methods are also applicable for interprocess communication protocols in operating systems. The process discovery methods use event data, in the form of an event stream collected from a software process execution. The goal is to infer a formal model of the behavior of the process. The goal is not a fully complete and correct process model, but a model with some formal description of the pattern of behavior that exist is in the process. A software engineer can then use this sketchy model to produce more complete models for his/her purposes. Cook and Wolf use finite automata as a notation.

As an inference method they use a modified version of Biermann and Feldman's k-tail heuristic [21]. Stroulia *et al.* [11] apply a grammatical inference algorithm to learn the services of Web applications based on usage scenarios. To integrate Web applications that offer information and services in the same domain, they have built a mediator that is responsible for interacting with the user and XML-speaking wrappers constructed for the applications. This approach shares the same downside as the applications used for behavioral modeling in the software design process: there is no way to know when the resulting synthesized model is complete, that is, when the set of source scenarios is covering enough. This is again due to the fact that the user does not know the desired model as a whole. This problem resembles the test coverage problem well known in software testing. Though the above modeling applications use inductive inference methods they can hardly be considered as design methods. Our fourth example of inductive methods is a genuine design method, which most likely has applications also in other similar tasks. Consider now the problem of synthesizing UML statechart diagrams from sequence diagrams (for UML concepts and notations the reader is referred to [5, 22, 23]).

A sequence diagram describes interactions among objects belonging to the system to be modeled. For synthesizing a statechart diagram for a particular object, all interactions concerning that object should be read from the sequence diagrams. The synthesis can be carried out by first linearizing the given set of sequence diagrams, that is, tranformed into strings over an alphabet describing the sent and received messages of the participating object. Since the statechart diagram is synthesized for one particant p at a time, the linearized strings are mapped into strings containing symbols involving the particant p only. The synthesis process is then completed by using an inference method that produces a statechart diagram consistent with the given sequence diagrams. It was already explained in [24] how the statechart synthesis process can be modeled as a language inference problem. The implementation of this idea is a system called SCED [8, 12]. Although regular languages are not inferable in the limit from positive data only, positive data can be sufficient if some additional conditions are applied. In SCED, the synthesizer seeks the statechart diagram with minimum number of states by using an exhaustive search method of Biermann and Krishnaswamy [25]. Minimizing the number of states in the resulting automaton is known to be NP-complete even when positive *and* negative data is available [26]. The main source of problems in SCED is that sequence diagrams do not give sufficient information about the behavior of system for inference. This often yields to "overgeneralized" statechart diagrams: the diagram also accepts additional scenarios to those given as input. In other words, programs are more "complete" than statechart diagrams because there is usually a valid continuation for every possible combination of variable values in every point of a program (except after the halt statement). In a statechart diagram, there is seldom a valid transition for every possible message in every state. Overgeneralization is considered the most severe problem of inductive inference from positive data, since only negative data can expose too general guesses [13]. The fact that the synthesized statechart diagram may generalize the given scenarios is often the desired effect, but in some cases the result is not what the user expects. If such harmful *implied scenarios* [27, 28] were allowed by the synthesis algorithm, the engineer should check the statechart diagram afterwards. If no support is given for the software designer, he/she should do the checking manually. In behavioral modeling of the software design process, this is not only inconvenient but also quite dangerous, because the resulting diagram can differ from desired one, but it might be difficult to notice this. However, the number of desired or harmless implied scenarios is typically larger than the number of harmful ones. Alur *et al.* [27] and Uchitel *et al.* [28] have proposed algorithms for detecting implied scenarios from the set of source scenarios, but these algorithms are meant for somewhat different situations than that appearing when grammatical inference is used as a design method. Namely, Alur *et al.* and Uchitel *et al.* consider situations where several statechart diagrams are combined to a single diagram modeling the behavior of a system instead of its separate participants.

It would be advantageous to have a synthesis algorithm that makes use of a stronger approach of grammatical inference than that used in SCED. Indeed, it is much more natural to consider the synthesis as a language inference problem under the MAT paradigm. In MAS (Minimally Adequate Synthesizer) algorithm [9, 10], the designer is the Teacher who is expected to answer the membership and equivalence queries posed by the inference algorithm that now plays the role of the synthesizer. Being a minimally adequate teacher requires that the designer can answer two kinds of simple questions:

1. the desinger must decide whether given sequences of messages are possible in the system to be imple-

mented (the membership queries),

2. the designer must accept or reject the statechart diagram proposed by the algorithm, and moreover, if it is rejected, a counterexample must be given (the equivalence queries).

Contrary to most algorithms for the statechart diagram synthesis problem, MAS is interactive allowing the user to guide the synthesis process. The user consultancy draws attention to key points in which the information expressed in the sequence diagrams is inaccurate or incomplete. Various aspects of implementing MAS are considered in [9, 10, 29, 30]. When designing the behavior of a software system, undesired generalizations in the state machine should be avoided. There is a risk of errors and inconsistencies if the designer is left to check and fix them afterwards. Thus, in this case, an algorithm like MAS seems a better choice than a fully automatic algorithm as the one used in SCED. Notice that in MAS, the non-existence of harmful implied scenarios in the resulting automaton depends on the designer's ability to answer the equivalence queries correctly. In order to help the designer in giving the correct answers, the user interface of the design tool should support the designer's decision making process. Different ways to do this in the case of MAS are discussed in [30]. Another problem in synthesizing UML statechart diagrams using MAS is the large number of membership queries. The algorithm tends to ask many membership queries that differ only slightly. Answering these questions could be frustrating in an unfavourable case because of the possibly large number of questions. A possible approach to manage this problem could be based on additional input information that would limit the number of membership queries. Such information could be given, for instance, in the form of pre and post conditions. Note that we have only considered input scenarios consisting of sent and received messages. However, the UML sequence diagram notation, and especially formal notations like Message Sequence Charts (MSC) [22], are much richer providing means to handle such additional information. In dynamic reverse engineering, however, the set of scenarios is used to capture the runtime behavior (e.g., containing method invocations, thrown exceptions, and constructor invocations) of the system during its usage. It is not reasonable to expect the user to be able to answer membership and equivalence queries when the example scenarios contain detailed nformation generated while running the (unknown) system. For such purposes, a fully automatic algorithm seems to be a better choice. Typically, in these applications the level of abstraction is lower than in "genuine" design applications allowing automatic inference methods to be applied. Hence, it seems that interactive methods suit for "design-by-example" processes while fully automatic means should be employed in "comprehension tools".

4 A design method

Using grammatical inference ts1606: voiko tuon poistaa: , especially MAT, in the synthesis problem has several advantages which seem to be independent from specific applications. This suggests that grammatical inference could be used as a general design method also in other applications. In this section we generalize our experiences with grammatical inference in the synthesis problem and describe its properties as a general design method. In the most general level of discussion, a design is usually divived into three stages: analysis, synthesis, and evaluation (for the general theory of design methods, see for example [31]). These three stages can be described as 'breaking the problem into pieces', 'putting the pieces together in a new way', and 'testing the new arrangement'. The fundamental feature of the design method we are proposing in this paper is *interaction* between the designer and the design tool. The designer is responsible for analyzing the problem. This is done by giving ts1606: use cases->usage scenarios usage scenarios which show how the resulting system should function. The synthesis stage is performed by the tool, which suggests a result by conjecturing a finite automaton. The designer evaluates the conjecture by accepting or rejecting it. A rejection causes a loop in which the stages are repeated until the evaluation results in acceptance. Although this approach is based on grammatical inference, it lacks many formal properties of grammatical inductive inference. Consider for example, criteria for successful inference. Instead of formal criteria, the success of the method depends on the subjective decision of the designer: whether or not he/she is satisfied with the resulting automaton as a model of the concept that is looked for. During the repetition loops the designer is allowed to change his/her mind when ideas become clearer. This makes it necessary for the tool used in the process, to go back and forth between various versions. in our case, too. The algorithm (the Learner) can pose questions cornerning the concept to be designed, which are currently unclear to the designer (the Teacher). The Learner might find it useful to give inaccurate answers (*Probably yes* and *Probaly no*) or to postpone an answer by saying *Later*. (These alternatives are discussed in [29] in the connection with MAS.) A sketch of a general MAT-based design tool could be given as follows:

```
begin
  receive sample use cases;
  repeat
    make necessary membership queries;
    output an automaton;
    if not accepted
      then receive a counterexample or
           perform the changes indicated by
           user's edit operations;
  until accepted;
  output the present automaton;
end;
```

Roughly speaking, this is the basic MAT algorithm augmented with the user's possibility to edit the automaton, and therefore to easily guide the inference process. We have earlier mentioned that the samples given by the designer are not arbitrary. On the other hand, the designer cannot provide the optimal set of samples, that is, the minimal set of samples defining the desired automaton. Such sets are known as *characteristic sets*. Oncina and García [32] have shown that characteristic sets for deterministic finite automata can be found in polynomial time (in a certain well-defined technical sense). However, this result does not help us because the designer does not exactly know the resulting automaton. The situation is analogous with that discussed in the connection with teaching. As mentioned earlier, it is possible that the user changes his/her mind or accidentally gives an incorrect answer to a query. However, it is reasonable that the design tool always operates as if all the information so far obtained were correct. Hence, we do not apply such concepts as "malicious omissions and errors" or the corresponding inference algorithms introducted by Angluin *et al.* in [33]. On the other hand, we cannot expect that the user is capable of providing (lexicographically) smallest counterexamples [34, 35], which would reduce the running time of the algorithm. This would again presume information about the resulting statechart that the Teacher (the designer) does not have. As mentioned earlier, if theTeacher had this information the inference process would be meaningless.

The main problem in the application of synthesizing UML statechart diagrams, and also in many other possible application areas, is to find the proper halting condition, that is, to recognize when the resulting automaton is complete. In our application, there cannot be any formal condition, since such a condition would imply that the resulting automaton is already known, making the use of the tool unnecessary. The concept of implied scenarios [27, 28] is one possible method for studying the problem of proper halting of a synthesis tool. An implied scenario exists because of a generalization made by the synthesis algorithm (and not by an explicit information given by the designer). Alur *et al.*'s [27] and Uchitel *et al.*'s [28] methods for detecting implied scenarios can be used to automatically warn the designer of the implied scenarios in the system. Detecting unwanted implied scenarios would cause a re-designing phase in the process.

Detecting the implied scenarios can also be use to help the user to evaluate the conjecture. For example, the tool could explicitly present some of the implied scenarios to the user so that it would be easier to decide whether the conjecture has the desided form. The implied scenarios presented to the user could be intelligently chosen to represent either the minimal implied scenarios (in the case of the first conjecture) or changes made compared the previous conjecture.

5 Conclusions and further work

We have discussed applications of grammatical inference in software engineering and introduced a design method which uses grammatical inference. Our main applications are in the area of software behavioral modeling, but we believe that the design method is applicable also elsewhere, even outside the scope of software engineering. We have suggested that inference methods allowing interaction are suitable for certain forward engineering tasks. On the other hand, "automatic" inference methods might fit better to reverse engineering tasks where the user has a limited knowledge of both the input scenarios and the desired system to be modeled and where the resulting automaton is mostly used for system comprehension only. Identification in the limit is a possible inference approach for these tasks. In both forward and reverse engneering, we have used positive examples only. From the grammatical inference point of view, the use of both positive and negative examples would make the process more efficient. However, the use of negative examples in the design process is somewhat contradicting: the designer is used to think how things should proceed, not what is undesirable. A possible way to obtain negative examples could be the use of methods to detect implied scenarios. As mentioned above, the existing methods of detecting implied scenarios are for combining automata, not for inferring one. Our further work concentrates on the problem of improving the interaction between the tool and the user, for example informing the user about the generalizations done by the algorithm. This makes it easier for the user to answer the equivalence queries, that is, to decide whether the conjectured automaton is the one he/she desires. Further goals are to restrict the number of membership queries by avoiding redundant/similar membership queries, and to find more "intelligent" and powerful ways to guide the algorithm. All these features would improve the algorithm by making it more efficient and more pleasant to use.

Acknowledgement

This work was supported by the Academy of Finland (Project 35025).

References

[1] Gold, E.M.: Language identification in the limit. Inform. Contr. **10** (1967) 447–474.

[2] García, P., Segarra, E., Vidal, E., Galliano, I.: On the use of morphic generator grammatical inference (MGGI) methodology in automatic speech recognition. International Journal of Pattern Recognition and Artificial Intelligence **4** (1990) 667–685.

[3] García, P., Vidal, E.: Inference of k-testable languages in the strict sense and applications to syntactic

pattern recognition. IEEE Trans. Pattern Anal. Mach. Intell. **PAMI-12** (1990) 920–925.

[4] Yokomori, T., Kobayashi, S.: Learning local languages and their application to DNA sequence. analysis. IEEE Trans. Pattern Anal. Mach. Intell. **PAMI-20** (1998) 1067–1079.

[5] Rumbaugh, J. Jacobson, J., Booch, G.: The Unified Modeling Language Reference Manual. Addison-Wesley, 1999.

[6] Cook, J.E.: Process Discovery and Validation through Event-Data Analysis. Ph.D. Dissertation, Dept. of Computer Science, University of Colorado, Boulder, Tech. Report CU-CS-817-96 (Nov. 1996).

[7] Cook, J.E., Wolf, A.L.: Discovering models of software processes from event-based data. ACM Trans. Softw. Eng. Meth. **7** (1998) 215–249.

[8] Koskimies, K., Männistö, T., Systä, T., Tuomi, J.: Automated support for modeling OO software. IEEE Softw. **15** (1998) 87–94.

[9] Mäkinen, E., Systä, T.: MAS – an interactive synthesizer to support behavioral modeling in UML. In: Proc. of International Conference on Software Engineering (ICSE'01), Toronto (May 2001) 15–24.

[10] Mäkinen, E., Systä, T.: Minimally adequate teacher synthesizes statechart diagrams. Acta Inform. **38** (2002), 235–259.

[11] Stroulia, E., Thomson, J., Situ, G.: Constrcuting XML-speaking wrappers for WEB applications: towards and interoperating WEB. In: Proc. of the 7th Working Conference on Reverse Engineering (WCRE 2000), Brisbane (Nov. 2000) 59–68.

[12] Systä, T.: Static and Dynamic Reverse Engineering Techniques for Java Software Systems. Ph. D. Dissertation, Dept. of Computer and Information Sciences, University of Tampere (May 2000).

[13] Angluin, D., Smith, C.H.: Inductive inference: theory and methods. ACM Comput. Surv. **15** (1983) 237–269.

[14] Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. **75** (1987) 87–106.

[15] Balcázar, J.L., Diaz, J., Gavaldà, R., Watanabe, O.: Algorithms for learning finite automata from queries: a unified view. In: Du, D.-Z., Ko, K.-I. (eds.): Advances in Algorithms, Languages, and Complexity, Kluwer Academic Publishers (1997) 73-91.

[16] Rivest, R.L., Schapire, R.E.: Inference of finite automata using homing sequences. Inf. Comput. **103** (1993) 299–347.

[17] Goldman, S.A., Mathias, H.D.: Teaching a smarter learner. J. Comput. Syst. Sci. **52** (1996) 255–267.

[18] Mathias, H.D.: A model of interactive teaching. J. Comput. Syst. Sci. **54** (1997) 487–501.

[19] Zhu, H.: A formal interpretation of software testing as inductive inference. Journal of Software Testing, Verification and Reliability **6** (1996) 3–31.

[20] Zhu, H., Hall, P., May, J.: Inductive inference and software testing. Journal of Software Testing, Verification and Reliability **2** (1993) 69–81.

[21] Biermann, A.W., Feldman, J.: On the synthesis of finite state machinces from samples of their behavior. IEEE Trans. Comput. **C-21** (1976) 592–597.

[22] Z.120 ITU-T Recommendation Z.120: Message Sequence Chart (MSC), ITU-T, Geneva, 1999.

[23] Object Management Group: OMG Unified Modeling Language Specification v.1.3, 20001 [<http://www.omg.org/>].

[24] Koskimies, K., Mäkinen, E.: Automatic synthesis of state machines from trace diagrams. Softw. Pract. Exper. **24** (1994) 643–658.

[25] Biermann, A.W., Krishnaswamy, R.: Constructing programs from example computations. IEEE Trans. Softw. Eng. **SE-2** (1976) 141–153.

[26] Gold, E.M.: Complexity of automaton identification from given data. Inform. Contr. **37** (1978) 302–320.

[27] Alur, R. Etessami, K., Yannakakis, M.: Inference of message sequence charts. In: Proc. of International Conference on Software Engineering (ICSE'00), Limerick, Ireland (2000) 304–313.

[28] S. Uchitel, S., Kramer, J., Magee, J.: Detecting implied scenarios in Message Sequence Chart specifications. In: Proc. of the 9th European Software Engineering Conferece and 9th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (ESEC/FSE'01), Vienna, Austria (Sept. 2001).

[29] Koskinen, J., Mäkinen, E., Systä, T.: Minimally adequate synthesizer tolerates inaccurate information during behavioral modeling. In: Proc. of SCASE'01, Enschede, The Netherlands (Feb. 2001).

[30] Koskinen, J., Mäkinen, E., Systä, T.: Implementing a component-based tool for synthesizing UML statechart diagrams. Acta Cybern. **15** (2002) 547–565,

[31] Jones, J.C.: Design Methods. Second Edition. Van Nostrand Reinhold, 1990.

[32] Oncina J., García, P.: Identifying regular languages in polynomial time. In: Bunke, H. (ed.): Advances in Structural and Syntactic Pattern Recognition (Proc. of the International Workshop on Structural and Syntactic Pattern Recognition) (1992) 99–108.

[33] Angluin, D., Kriķis, M., Sloan, R.H., Turán, G.: Malicious omissions and errors in answers to membership queries. Mach. Learn. **28** (1997) 211–255.

[34] Birkendorf, A., Böker, A., Simon, H.U.: Learning deterministic finite automata from smallest counterexamples. In: Proc. 9th ACM/SIAM Symp. Discr. Alg. (SODA) (Jan. 1999) 599–608.

[35] Ibarra, O.H., Jiang, T.: Learning regular languages from counterexamples. J. Comput. Syst. Sci. **43** (1991) 299–316.

An Extension of the Bellman-Ford Algorithm for QoS Routing with Inaccurate Information

Alpár Fancsali
Department of Telecommunications, Budapest University of Technology and Economics (BME),
Magyar tudósok körútja 2., 1117 Budapest, Hungary
E-mail: alpar@hit.hit.bme.hu

Keywords: QoS routing, Bellman-Ford algorithm, abstract algebra

Received: April 29, 2002

This paper investigates QoS routing in IP networks. The major concern is the real-time selection of paths to fulfil distinct quality of service requirements (such as end-to-end delay or bandwidth requirements). In practice one has to tackle routing with inaccurate information when link measures are subject to random fluctuations described by some given p.d.f.-s. Our goal is to identify a path that is most likely to successfully accommodate the desired QoS parameter. Unfortunately, this task is generally NP-hard. In the paper this problem is attacked by an abstract algebraic extension of the Bellman-Ford shortest path selection algorithm. The paper demonstrates that this extended algorithm is still tractable and gives a very good approximation for the optimal solution of the original NP-hard task.

1 Introduction

One of the major challenges in IP networking is to ensure QoS routing, which selects paths to fulfil given quality of service requirements, for example end-to-end delay or bandwidth requirements [1, 2]. Because of the manifold optimization criteria, QoS routing is much harder than the problem of finding optimal paths based merely on the hop count [3]. Papers [4, 5] investigate in detail the fulfilment of the bandwidth requirement, therefore in the further discussion this paper deals with ensuring only the end-to-end delay specification, which is an NP-hard task. The OSPF standard offers two (and PNNI even more) levels, where routing can be performed in a hierarchical manner [6]. Subnetworks on a given level of the hierarchy are abstracted as "nodes" for a higher layer and the delay information in those subnetworks is aggregated into an average QoS parameter. On the other hand, randomly fluctuating traffic load on links can also result in random delays. Thus, link delays are periodically measured in the network. When the delay surpasses a given threshold the link advertises this threshold[1]. These thresholds are defined in advance. This prompts us to take delays into account as random variables characterized by their probability distribution functions over the interval between two reported thresholds [4, 8]. The distribution of these thresholds can be either equidistant or non-equidistant. In practice the latter one is used, since in this case the impact on network utilization by sending only signaling information can be minimized. (Considering that a part of the available bandwidth is used for carrying information about link delays.)

The phenomena described above give rise to the task of routing with inaccurate information. Namely, how to select paths to fulfil end-to-end delay requirements in the lack of the exact values of link delays [1, 4, 8]. Then routing is perceived as an optimization problem, where among the different quality paths one is selected, which meets the end-to-end QoS requirement with maximal probability. Unfortunately, routing with inaccurate information in general cannot be reduced to the well-known Shortest Path Routing (SPR), that is why this paper suggests a novel method for solving the task.

The results are discussed in the following structure:

- In *Section 2* the underlying model is introduced and brief summary of notations and some initial results are given.
- *Section 3* gives an abstract algebraic extension of the well-known Bellman-Ford algorithm, which can be applied to perform the QoS routing with inaccurate information.
- *Section 4* gives a binary algebra over which the above generalized Bellman-Ford algorithm is able to work.
- In *Section 5* the developed method is treated in the case of three special link delay distribution models.
- *Section 6* mentions the approximative nature of the method and gives an analytical estimation for the error of the generalized Bellman-Ford (GBF) algorithm.
- Finally, *Section 7* contains an extensive numerical analysis, which demonstrates that the QoS routing

[1] This mechanism was specified only for the bandwidth QoS descriptor in the QOSPF standard [6, 7]. However, this standard will be probably extended to the additive QoS descriptors (e.g. to the quite relevant end-to-end delay).

with inaccurate information can be efficiently performed by the presented GBF algorithm in polynomial time.

2 Additive routing with inaccurate information - the model

To model the routing problem let the following quantities be given:

- there is a weighted graph $G(V, E, C)$ representing the network topology, where nodes are denoted by index $u \in V$, links are referred to as node pairs $(u, v) \in E$, and function $C(R)$ gives the overall weight of the edge set $R \subset E$ ($C(R)$ is exactly defined in the next section);
- each link $(u, v) \in E$ has an additive QoS descriptor $\delta_{(u,v)}$ (e.g. delay) which is assumed to be a random variable subject to a probability distribution function $F_{(u,v)}(t) = P\left(\delta_{(u,v)} \leq t\right)$ (or a probability density function $f_{(u,v)}(t) := dF_{(u,v)}(t)/dt$, the value set of which includes the Dirac delta in order to describe discrete distributions, too);
- link delays are supposed to be independent random variables;
- a path is described by an edge set $R = \{(v_0, v_1), (v_1, v_2), \ldots, (v_{L-1}, v_L)\}$, where L is the number of hops, $v_i \in V$ for $i = 0, 1, \ldots, L$ (R stands for a path connecting a predefined source node $v_0 = s$ and a destination node $v_L = f$);
- set $\mathcal{R}_{sf}$ denotes the set of all the paths between source node s and destination node f;
- there is a maximum delay requirement D for a path R (i.e. $\sum_{(u,v)\in R} \delta_{(u,v)} \leq D$ is expected).

The objective is to find an optimal path R_{opt} which is most likely to fulfil the given QoS criterion, namely:

$$R_{\text{opt}} : \max_{R\in\mathcal{R}_{sf}} P\left(\sum_{(u,v)\in R} \delta_{(u,v)} \leq D\right). \qquad (1)$$

The path R_{opt} introduced above will be referred to as the Most Likely Path (MLP). Task (1) is considered the additive routing with inaccurate information (ARII).

Lemma 1 (Guérin et al) *The problem of ARII (1) is NP-hard.*

The proof is based on the fact that the problem $\mathbf{P}(\pi)$: $\tilde{R} : P\left(\sum_{(u,v)\in R} \delta_{(u,v)} < D\right) > \pi$ (where $0 < \pi < 1$ is some given threshold) is NP-hard, because an instance of the Kth largest subset problem (known to be NP-hard, see [9]) can be transformed into a special case of problem $\mathbf{P}(\pi)$. (For more details see [4].)

In paper [4] the authors reduced ARII into the tractable bottleneck type routing, utilizing that end-to-end delay on a path can be approximated under certain scheduling scenarios (such as Weighted Fair Queuing Scheduler, Rate Controlled Earliest Deadline First Schedulers). Unfortunately, these approximations (so-called rate based bounds) are rather crude, so they cannot reach adequate quality. Thus, one has to find alternative solutions to make ARII tractable. Paper [4] also offers several heuristic methods, but the hop count of paths always remains in the formulas, which makes the algorithms difficult to deal with. Another way to make ARII tractable is demonstrated by [5, 10] where the authors reduced the MLP selection into shortest paths problem by applying tail distribution estimation (Chernoff inequality). Although the efficiency of this technique is based on simulation results that proved to be very good, there is no theoretical consideration which accounts for the deviation between the probability of the optimal MLP by (1) and the probability of the path given by the algorithm.

The next section proposes a method which addresses the original problem directly. To this, one has to look for the simplest algebraic structure over which the traditional shortest path selection algorithms (Dijkstra, Bellman-Ford [11]) are able to work and provide optimal solution. The detailed treatment will be demonstrated on the Bellman-Ford algorithm, because it is capable of distributed operation on the network nodes, which considerably simplifies the routing protocol.

3 An abstract algebraic generalization of the Bellman-Ford algorithm

This section proposes a general description of the shortest path routing, where the weights take their values not only from the set of real numbers, but from arbitrary basic sets. However, in this case it is rather ambiguous what the shortest path means. Hence, the notion of the shortest path needs to be defined first.

3.1 The generalized shortest path

Let the following quantities be given:

- a set A, an element of which corresponds to some features of a link (u, v) (denoted by $c_{uv} \in A$) or of a path R (denoted by $C(R) \in A$);
- a binary operator $\beta(a, b)$ on set A (i.e. $\beta(a, b) \in A$ for all $a, b \in A$) which is chosen in such a way that if $R = R_1 \cup \overline{R}_1$, then $C(R_1 \cup \overline{R}_1) = \beta\left(C(R_1), C(\overline{R}_1)\right)$, where R_1 is a subpath of R and $\overline{R}_1 = R \backslash R_1$;

– a relation $\leq^*$ which orders the elements of set A according to a predefined performance criterion (e.g, if a path R_1 is not worse than path R_2 under some consideration, then $C(R_1) \leq^* C(R_2)$);

Definition 1 *The binary algebra $[A, \beta]$ is called monoid [12] if β is associative ($\beta(\beta(a,b),c) = \beta(a,\beta(b,c))$ for all $a, b, c \in A$) and if set A contains identity element ($1 \in A$), then $\beta(a,1) = \beta(1,a) = a$ for all $a \in A$.*
A monoid is commutative if β is commutative, namely $\beta(a,b) = \beta(b,a)$ for all $a, b \in A$.

After this, one can stipulate that the "cost" of the empty path corresponds to the identity element in A, i.e. $C(\emptyset) = 1 \in A$. Let us introduce a notation for applying β successively n-times (similarly to symbol Σ):

$$\overset{n}{\underset{i=1}{B}} x_i \equiv \beta(...(\beta(...(\beta(\beta(x_1,x_2),x_3)...),x_i)...),x_n) \quad x_i \in A, \quad i = 1,2,...,n. \tag{2}$$

Furthermore, one can observe that relation $\leq^*$ uniquely defines the notation "min", because

$$x_{min} = \min{}^*\{x : x \in X, X \subset A\} \equiv \exists x_{min} \in X \subset A : x_{min} \leq^* x \quad \forall x \in X. \tag{3}$$

If relations $x_1 \leq^* x_2$ and $x_2 \leq^* x_1$ can hold at the same time, while $x_1 \neq x_2$, then x_1 is equivalent to x_2, which is denoted as $x_1 \sim x_2$.

It is obvious that if relation $a \leq^* b$ does not hold, a and b cannot be equal. In this case one can use the notation $b <^* a$: $\neg(a \leq^* b) \equiv b <^* a$.

Based on these notations, the definition of the generalized shortest path can be given as follows:

Definition 2 *The generalized shortest path of a graph $G(V,E,C)$ connecting source node s and destination node f is obtained by solving the following optimization task:*

$$R_{\text{opt}} : \min_{R \in \mathcal{R}_{sf}}{}^* \underset{(u,v)\in R}{B} c_{uv}, \quad c_{uv} \in A, \tag{4}$$

where c_{uv} characterizes link (u,v) and $C(R)$ characterizes the path R, $C(R) = B_{(u,v)\in R} c_{uv}$.

When $A = \mathbf{R}^+$ and $\beta(a,b) = a+b$ ($a, b \in \mathbf{R}^+$), furthermore $\leq^*$ is the traditional $\leq$ on the set of real numbers ($\mathbf{R}$), the optimization formula (4) corresponds to the traditional shortest path routing (which can be performed by Dijkstra's algorithm with $O(N^2)$ complexity or the Bellman-Ford algorithm with $O(N^3)$ complexity).

3.2 The generalized Bellman-Ford (GBF) algorithm

We look for the optimal path R_{opt} from a source node $s \in V$ to a destination node $f \in V$ in a graph. Assign a pair to each node $v \in V$ as $Q_v = (z_v, d_{vf})$ where the so-called "parent node" z_v gives the neighbouring node to which direction the optimal path (from v to f) is built at the end of the algorithm ($z_v \in V$ and $(v, z_v) \in E$), and $d_{vf} \in A$ characterizes this path from v to f via z_v. Define an infinity element in A having the following properties: $a \leq^* \infty$ and $\beta(\infty, a) = \beta(a, \infty) = \infty$ for all $a \in A$. Then the algorithm is defined by the following steps:

1. Initialize the labels Q_v by setting $z_v[0] := v$ for all $v \in V$, furthermore $d_{vf}[0] := \infty$ if $v \neq f$ and $d_{ff}[0] := 1$ [2]. Then perform the following steps for all $v \in V \backslash \{f\}$ (which can be carried out in parallel at every node).

2. Assume that we are in step k. Send label $d_{vf}[k]$ (from $Q_v[k]$) to all the neighbouring nodes; the set of which is denoted by $T_v = \{w : (v,w) \in E\}$. (As each node acts in the same manner, elements d_{wf} ($w \in T_v$) are available at node v.)

3. Let $u \in T_v$ be a neighbouring node for which the following equation holds:

$$\beta(c_{vu}, d_{uf}[k]) = \min_{w \in T_v}{}^* \beta(c_{vw}, d_{wf}[k]).$$

   If $\beta(c_{vu}, d_{uf}[k]) <^* d_{vf}[k]$, overwrite Q_v as $d_{vf}[k+1] := \beta(c_{vu}, d_{vf}[k])$ and $z_v[k+1] := u$, otherwise $Q_v[k+1] := Q_v[k]$.

4. If $Q_v[k+1] = Q_v[k]$ for all $v \in V$ or $k = |V| - 2$ then STOP, otherwise $k := k + 1$ and go to 2.

At the termination of the algorithm, choosing any starting node $s \in V \backslash \{f\}$, the parent nodes z_v in Q_v uniquely determine a path from s to f, provided that such path exists:

$$R^* = \{(v_0, v_1), (v_1, v_2), \ldots, (v_{L-1}, v_L)\},$$

where $v_0 = s$, $v_L = f$ and $v_i = z_{v_{i-1}}$, $i = 1, 2, \ldots, L$.

Furthermore, there is a need to investigate under what conditions the path R^* obtained by the GBF algorithm is the solution of the optimization formula posed in (4). Namely, under what conditions the expression $C(R_{\text{opt}}) \sim C(R^*)$ holds.

Theorem 1 *The GBF algorithm defined over $[A, \beta]$ terminates at most in $2I(|V|-1)|E|$ steps (I is a constant) if monoid $[A, \beta]$ fulfils the following three conditions:*

C1 *operation $\beta(a,b)$ is associative and commutative;*

C2 *relation $\leq^*$ is transitive, i.e. if $a \leq^* b$ and $b \leq^* c$ then $a \leq^* c$;*

C3 *$a \leq^* \beta(a,b)$ for all $a, b \in A$ (*non-negativity*),*

Constant I equals to the computation demand for evaluating β and $\leq^*$. Furthermore, if

[1] In case of the traditional BF algorithm the label ordered to the destination node was zero so the choice $d_{ff}(0) = 1$ seems unaccustomed. However, do not forget that the identity of $[\mathbf{R}^+, +]$ is the mere zero, because $a + 0 = 0 + a = a$ for all $a \in A$.

C4 $\beta(a,c) \leq^* \beta(b,c)$ *if* $a \leq^* b$ *for all* $a,b,c \in A$ *(*monotonity*)*

is also satisfied, then the GBF algorithm provides optimal solution. Namely $C(R^*) \sim C(R_{\rm opt})$, R^* *is* $R_{\rm opt}$ *or equivalent to* $R_{\rm opt}$.

The proof is fully spelt out in the Appendix.

The next section shows how to apply this algorithmic tool for ARII.

4 MLP selection in the case of given p.d.f.-s

Now we will find an algebraic structure over which the GBF algorithm is capable of operating and provides the solution of the initial problem (1) (at least in an approximate fashion). The task of MLP selection can be transformed into the following form, using the complement probability:

$$R_{\rm opt} : \min_{R \in \mathcal{R}_{sf}} P\left(\sum_{(u,v)\in R} \delta_{(u,v)} > D \right). \tag{5}$$

Let A denote the set of probability density functions with domain $\mathbf{R}^+$ as

$$A = \left\{ f(x) : \begin{array}{l} \int_0^\infty f(x)dx = 1, \\ f(x) = 0, \text{ if } x < 0, \\ f(x) \geq 0 \ \forall x \in \mathbf{R} \end{array} \right\}. \tag{6}$$

Since each link $(u,v) \in E$ is described by a density function, here

$$c_{uv} \equiv f_{(u,v)}(x) := \frac{dP\left(\delta_{(u,v)} \leq x\right)}{dx} \in A.$$

Then function $C(R)$ of the graph $G(V,E,C)$ denotes the density function of the overall delay measured along a path R. Because links are assumed to be independent random variables, let operation β be the convolution, namely:

$$\begin{array}{c} \beta(f,g)(z) = f * g(z) = \\ \int_0^z f(x)g(z-x)dx = \int_0^z g(x)f(z-x)dx \end{array} \tag{7}$$

and

$$C(R,x) = \left(\underset{(u,v)\in R}{B} f_{(u,v)} \right)(x). \tag{8}$$

According to (5), it is known that a path R_1 is not worse than a path R_2 if $P\left\{ \sum_{(u,v)\in R_1} \delta_{(u,v)} > D \right\} \leq P\left\{ \sum_{(u,v)\in R_2} \delta_{(u,v)} > D \right\}$. Thus, one can define the relation $\leq^*$ as follows:

$$\begin{array}{c} C(R_1) \leq^* C(R_2) \Leftrightarrow \\ \int_D^\infty C(R_1,x)dx \leq \int_D^\infty C(R_2,y)dy. \end{array} \tag{9}$$

With the notations introduced above, (5) can be rewritten just like (4), namely

$$R_{\rm opt} : \min_{R\in\mathcal{R}_{sf}}{}^* \underset{(u,v)\in R}{B} f_{(u,v)}. \tag{10}$$

If monoid $[A,\beta]$ and relation $\leq^*$ fulfilled the conditions C1-C4 in Theorem 1, task (10) could exactly be completed by the GBF algorithm described in Section 3.2. It is obvious that relation $\leq^*$ is transitive and operation β (the convolution) is associative and commutative (i.e. C1,C2 are satisfied).

Lemma 2 *The monoid* $[A,\beta]$ *meets the non-negativity criterion (C3) with the relation* $\leq^*$.

Proof:
According to the introduced notations, non-negativity could be formulated as follows:

$$\begin{array}{c} f \leq^* f * g \ \equiv \\ \int_D^\infty f(x)dx \leq \int_{z=D}^\infty \int_{y=0}^z f(y)g(z-y)dydz. \end{array} \tag{11}$$

for all $f,g \in A$. To prove the validity of (11), take a lower bound of the right hand side of the inequality, narrowing down the integral interval:

$$\begin{array}{c} \int_{z=D}^\infty \int_{y=0}^z f(y)g(z-y)dydz \geq \\ \int_{z=D}^\infty \int_{y=D}^z f(y)g(z-y)dydz. \end{array} \tag{12}$$

As g is a function with domain $\mathbf{R}^+$ (i.e. $g(z-y) = 0$ if $y > z$), one can rewrite (12) as follows:

$$\begin{array}{c} \int_{z=D}^\infty \int_{y=D}^\infty f(y)g(z-y)dydz = \\ \int_{y=D}^\infty f(y) \int_{z=D}^\infty g(z-y)dydz, \end{array} \tag{13}$$

where $\int_{z=D}^\infty g(z-y)dz = \int_{z=y}^\infty g(z-y)dz = \int_0^\infty g(q)dq = 1$, which implies that (13) equals to $\int_D^\infty f(y)dy$. This proves the statement of the lemma. ■

Unfortunately, the condition C4 (monotonity) does not hold in general, as the following simple example illustrates. Consider three discrete distributions given by Dirac impulses as follows:

$$\begin{array}{l} a(x) = 0.32\delta(x) + 0.64\delta(x-1) + 0.04\delta(x-2) \\ b(x) = 0.66\delta(x) + 0.08\delta(x-1) + 0.26\delta(x-2) \\ c(x) = 0.27\delta(x) + 0.5\delta(x-1) + 0.23\delta(x-2). \end{array}$$

If $D = 1.5$ then $a \leq^* b$, because $\int_D^\infty a(x)dx = 0.04$ and $\int_D^\infty b(x)dx = 0.24$. However,

$$\begin{array}{ll} a * c = & 0.09\delta(x) + 0.34\delta(x-1) + 0.4\delta(x-2) + \\ & 0.16\delta(x-3) + 0.01\delta(x-4) \\ b * c = & 0.18\delta(x) + 0.35\delta(x-1) + 0.26\delta(x-2) + \\ & 0.15\delta(x-3) + 0.06\delta(x-4) \end{array}$$

and $0.4 + 0.16 + 0.01 > 0.26 + 0.15 + 0.06$, namely the inequality $\beta(a,c) \leq^* \beta(b,c)$ is not satisfied, C4 does not hold.

However, it is no wonder that one or more conditions need to be violated because otherwise a polynomial algorithm would solve an NP-hard problem. Fortunately, conditions C1-C3 hold, so the GBF algorithm can be applied as an approximate method. In Section 6 it is also analysed how much error the violation of monotonity causes in some important, practical cases. As pointed out by numerical analysis this will not impair the performance of the method.

5 Extending the QOSPF routing to MLP selection

Link delays are advertised to the nodes in each time instant. (This time instant is set up by the network operator.) In order to reduce the bandwidth used for carrying information about link delays, the values of delay are quantized in the following manner:

- The delay axis (the set of possible values of the link delays) is covered with a grid $\tau = \{t_i, i = 1, 2, \ldots, Z\}$.
- At each time instant link (u,v) advertises the last t_i value its link delay has exceeded, which implies that $\delta_{(u,v)} \in (t_i, t_{i+1})$.

For the sake of simplicity, intervals (t_i, t_{i+1}) are firstly assumed to have same length Δ (equidistant distribution), namely: $t_i - t_{i+1} = \Delta$ for all $i = 1, 2, \ldots, Z$. Let $d_{(u,v)} \in \tau$ denote the delay threshold currently advertised by link (u,v). Then one can introduce a random variable $\gamma_{(u,v)}$ as $\gamma_{(u,v)} = \delta_{(u,v)} - d_{(u,v)}$ which takes its values from the interval $(0, \Delta)$. Since the appropriate distributions for the local link delays are not exactly known, the modeller has some liberty to choose any p.d.f., which leads to a realistic model. (Instead of identifying the appropriate distributions, this paper focuses on making the NP-hard ARII tractable.) This section proposes three realistic models (with using uniform, Bernoulli and normal distributions).

5.1 Uniform distribution model

Let us suppose that link delays take their values uniformly from the current delay interval. More precisely,

$$P(\gamma_{(u,v)} < x) = \begin{cases} x/\Delta, & \text{if} \quad 0 \leq x \leq \Delta \\ 0, & \text{if} \quad x < 0 \\ 1, & \text{if} \quad x > \Delta. \end{cases} \tag{14}$$

To calculate the "probability of a path R", one can take the convolution of (14) with itself successively $L-1$ ($L = |R|$) times. From this, the following distribution function is obtained [13]:

$$F_e(x,L) = \begin{cases} 0, & \text{if} \quad x \leq 0 \\ 1, & \text{if} \quad x \geq L \cdot \Delta \\ \frac{1}{L!} \sum\limits_{i \geq 1} (-1)^i \begin{pmatrix} L \\ i \end{pmatrix} \left(\left\lceil \frac{x}{\Delta} - i \right\rceil^+ \right)^L, & \\ & \text{otherwise,} \end{cases} \tag{15}$$

where $\lceil z \rceil^+ = z$ if $z > 0$, otherwise $\lceil z \rceil^+ = 0$. By using function $F_e(x,L)$ the "probability of a path R" is $P\left(\sum_{(u,v)\in R} \delta_{(u,v)} < D\right) = F_e\left(D - \sum_{(u,v)\in R} d_{(u,v)}, |R|\right)$. From this, one can observe that the sum $d_\Sigma = \sum_{(u,v)\in R} d_{(u,v)}$ and the hop count $L = |R|$ uniquely characterize the path R. This is why one can define the monoid $[A, \beta]$ and relation $\leq^*$ in the following manner:

- Let the basic set A be defined as the set of the possible pairs (d_Σ, L):

  $$A = \left\{ (d_\Sigma, L) | d_\Sigma \in \mathbf{R}^+, L \in \mathbf{N}^+ \right\}.$$

  (The indexed variables a_{d_Σ} and a_L will refer to the parts of an element $a \in A$.) A link (u,v) is characterized by the element $c_{uv} = (d_{(u,v)}, 1)$.
- The operation of the convolution on set A is interpreted as $\beta(a,b) = (a_{d_\Sigma} + b_{d_\Sigma}, a_L + b_L)$ for all $a, b \in A$.
- When $F_e\left(D - \sum_{(u,v)\in R_1} d_{(u,v)}, |R_1|\right) \geq F_e\left(D - \sum_{(u,v)\in R_2} d_{(u,v)}, |R_2|\right)$ for the paths R_1, R_2, then path R_1 is not worse than path R_2. Therefore $\leq^*$ is chosen as follows:

  $$a \leq^* b \Leftrightarrow 1 - F_e\left(D - a_{d_\Sigma}, a_L\right) \leq 1 - F_e\left(D - b_{d_\Sigma}, b_L\right). \tag{16}$$

Due to Lemma 2, the GBF algorithm over the monoid $[A, \beta]$ with the relation $\leq^*$ defined above is stable.

Because of the complexity of the function $F_e(x,L)$ (15), evaluating $\leq^*$ takes a relatively long time. That is why the next subsection proposes a simpler model.

5.2 On-off model

In order to simplify the computations the random delays are assumed to be binary random variables described by the following distribution:

$$P(\gamma_{(u,v)}^{\text{on-off}} = 0) = 1 - \tfrac{m_{(u,v)}}{\Delta}, \quad P(\gamma_{(u,v)}^{\text{on-off}} = \Delta) = \tfrac{m_{(u,v)}}{\Delta}$$

where $m_{(u,v)} = \mathbf{E}\left\{\gamma_{(u,v)}\right\}$. This model is referred to as "on-off model" in traffic engineering. Based on the "on-off" model, the following property holds:

Let $\gamma_1, \gamma_2, \ldots, \gamma_L$ be independent, identically distributed (i.i.d.) random variables which take their values from the interval $[0, \Delta]$. Then for all $x \in \mathbf{R}$

$$P(\gamma_1 + \gamma_2 + \ldots + \gamma_L > x) \leq \\ \leq P(\gamma_1^{\text{on-off}} + \gamma_2^{\text{on-off}} + \ldots + \gamma_L^{\text{on-off}} > x). \quad (17)$$

(The proof can be found in [14].) Applying this property and taking into account that the sum of L independent random variables has a binomial distribution, one can give a lower bound for the distribution $F(x, L) = P(\sum_{(u,v)\in R} \gamma_{(u,v)} < x)$ as follows:

$$F(x, L) \geq F_{\text{on-off}}(x, L) = \\ \sum_{k<x/\Delta} \left(\begin{array}{c} L \\ k \end{array} \right) \left(\frac{m}{\Delta}\right)^k \left(1 - \frac{m}{\Delta}\right)^{L-k}. \quad (18)$$

Returning now to the GBF algorithm described in Section 3.2, we can provide a fast routing algorithm by using the "on-off" approximation, when $\leq^*$ is defined by $F_{\text{on-off}}(x, L)$ instead of the function $F_e(x, L)$ given in (16). (The underlying monoid $[A, \beta]$ remains the same as in the previous subsection.) Considering that the array $\mathbf{F}_{\text{on-off}} =$

$$\left[\sum_{k=0}^{q} \left(\begin{array}{c} l \\ k \end{array} \right) \left(\frac{m}{\Delta}\right)^k \left(1 - \frac{m}{\Delta}\right)^{l-k} \right]_{\substack{l=0,1,\ldots,N-1 \\ q=0,1,\ldots,l}}$$

can be calculated and stored beforehand, the distribution of an L-hop path reduces to

$$F_{\text{on-off}}(x, L) = \begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{if } x \geq L\Delta \\ (\mathbf{F}_{\text{on-off}})_{L, \text{int}\left[\frac{x}{\Delta}\right]}, & \text{otherwise.} \end{cases} \quad (19)$$

This implies that the running time of the GBF algorithm over $[A, \beta]$ basically equals to the running time of the traditional BF algorithm (over $[\mathbf{R}^+, +]$).

Of course, the "probability of a path" is not exactly determined by the "on-off" delay model, which implies the approximative nature of the result. However, due to the property (18), this yields a worst case analysis which is important in engineering. If $\gamma_{(u,v)}$-s are uniformly distributed on $(0, \Delta)$, then $m = \mathbf{E}\left\{\gamma_{(u,v)}\right\} = \Delta/2$ in (18).

5.3 Normal distribution model

If $\gamma_{(u,v)}$-s are i.i.d. random variables, the sum $\sum_{(u,v)\in R} \gamma_{(u,v)}$ approximately has a normal distribution according to the central limit theorem. This approximation is rather accurate in case of paths having more than 3-4 hops. Thus, it seems realistic to assume that even local link delays are normally distributed. On the other hand the modeller has some liberty to choose any p.d.f.-s in order to simplify the computation, because the concrete delay is unknown in the interval $\delta_{(u,v)} \in (t_i, t_{i+1})$. With respect to the underlying computation in the GBF algorithm the normal distribution assumption is promising, as the convolution of normal distributions can be quite simply evaluated, because we only have to deal with the expected value and the standard deviation. The greatest advantage of this model is that it does not demand equidistant link delay advertisement intervals in τ (as was required in sections 5.1, 5.2). This subsection will discuss the ARII in case of arbitrary, non-equidistant grids.

Suppose that $\delta_{(u,v)} \in (t_i, t_{i+1})$ where $t_i \in \tau$ is the latest advertised threshold. Despite the fact that a normal distribution is defined over an infinitely long interval, one can fit a normal distribution $N(m, \sigma)$ over a finite interval with an ϵ error by solving the following approximation task:

$$\sigma_{(u,v)}: \int_{t_i}^{t_{i+1}} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(x-m_{(u,v)})^2}{2\sigma^2}} dx = 1 - \epsilon, \\ \text{where } m_{(u,v)} := \frac{t_{i+1}+t_i}{2} \quad (20)$$

(ϵ will be the probability of $\delta_{(u,v)} \notin (t_i, t_{i+1})$). According to the normal distribution model, a link (u, v) is characterized by the function $\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \exp(x^2/2) dx$ in the following fashion:

$$P(\delta_{(u,v)} \leq x) = \Phi\left(\frac{x - m_{(u,v)}}{\sigma_{(u,v)}}\right).$$

Since $\delta_{(u,v)}$ is assumed to have a normal distribution, the overall path delay $\sum_{(u,v)} \delta_{(u,v)}$ is also normally distributed with expected value $m(R) = \sum_{(u,v)\in R} m_{(u,v)}$ and with variance $\sigma(R) = \sqrt{\sum_{(u,v)\in R} \sigma^2_{(u,v)}}$. This is due to the fact that link delays are assumed to be independent random variables and the convolution of normal distributions is also normal. Therefore,

$$P\left(\sum_{(u,v)\in R} \delta_{(u,v)} \leq D\right) = \Phi\left(\frac{D - m(R)}{\sigma(R)}\right). \quad (21)$$

Then the following theorem holds.

Theorem 2 *Define the monoid $[A, \beta]$ and the relation $\leq^*$ as*

- *$A := \{(m, \sigma) : m, \sigma \in \mathbf{R}^+\}$ (the indexed variables a_m, a_σ will refer to the parts of an element $a \in A$; then a link (u, v) is characterized by $c_{uv} := (m_{(u,v)}, \sigma_{(u,v)})$;*
- *$\beta(a, b) = \left(a_m + b_m, \sqrt{a_\sigma^2 + b_\sigma^2}\right)$ for all $a, b \in A$;*
- *$a \leq^* b \Leftrightarrow b_\sigma \lceil D - a_m \rceil^+ \geq a_\sigma \lceil D - b_m \rceil^+$.*

Then the GBF algorithm operating over $[A, \beta]$ with the relation $\leq^*$ solves ARII according to the normal distribution model under the condition that the path selection is restricted to paths having at least 0.5 probability (i.e. $P\left(\sum_{(u,v)\in R} \delta_{(u,v)} \leq D\right) \geq 0.5$).

Proof:
Since β is associative and commutative and $\leq^*$ is transitive, to use the GBF algorithm as an approximative method only the non-negativity needs to be proven. To do this, one has to show that

$$\begin{gathered} a \leq^* \beta(a,b) \equiv \\ \sqrt{b_\sigma^2 + a_\sigma^2}\,\lceil D - a_m \rceil^+ \geq a_\sigma \lceil D - a_m - b_m \rceil^+ \end{gathered} \tag{22}$$

for all $a, b \in A$. As variances and expected values are positive, it is obvious that $\sqrt{b_\sigma^2 + a_\sigma^2} > a_\sigma$ and $\lceil D - a_m \rceil^+ \geq \lceil D - a_m - b_m \rceil^+$, which implies that (22) is satisfied. One can note that the non-negativity would not be fulfilled in general without using $\lceil . \rceil^+$ ($\lceil z \rceil^+ = z$ if $z > 0$, otherwise $\lceil z \rceil^+ = 0$). This restriction implies that the paths having less than 0.5 probability cannot be distinguished. Finally, we show that if a path R_1 is not worse than a path R_2, namely

$$\Phi\left(\frac{D - m(R_1)}{\sigma(R_1)}\right) \geq \Phi\left(\frac{D - m(R_2)}{\sigma(R_2)}\right), \tag{23}$$

then $C(R_1) \leq^* C(R_2)$, provided R_1, R_2 have at least 0.5 probability. Due to the monotonity of function $\Phi(x)$, (23) can be rearranged to the following form:

$$\sigma(R_2)\,(D - m(R_1)) \geq \sigma(R_1)\,(D - m(R_2)). \tag{24}$$

From the assumption that R_1, R_2 have at least 0.5 probability, we know that $m(R_1) \leq D$ and $m(R_2) \leq D$. Therefore, the use of brackets $\lceil . \rceil^+$ instead of $(.)$ in (24) does not cause further restriction. This completes the proof. ■

It is worth noting that the assumption $m(R) < D$ does not mean a real restriction, because in practice a path having greater $m(R)$ than the delay requirement D is unacceptable.

5.4 Applying an improved relation

It often occurs that $C(R_1) \sim C(R_2)$ (e.g. when both paths meet the delay requirements with zero or one probabilities). Due to the experiences, the GBF algorithm yields more exact solutions by selecting the path having the smallest expected value among the others being equivalent with each other. Let $\leq'$ denote this new relation in the following way:

$$\begin{gathered} C(R_1) \leq' C(R_2) \Leftrightarrow \\ [C(R_1) <^* C(R_2)] \quad \vee \quad [C(R_1) \sim C(R_2)] \wedge \\ \left[\sum_{(u,v)\in R_1} \mathbf{E}\delta_{(u,v)} \leq \sum_{(u,v)\in R_2} \mathbf{E}\delta_{(u,v)}\right]. \end{gathered} \tag{25}$$

This modification does not influence the results obtained so far, because the comparison principle related to the tail distribution has not been changed.

6 The error of the GBF algorithm

Uptil now we have not dealt with the error caused by violating condition C4 in Theorem 1 (monotonity). In this section the difference between the probability of the path R^* given by the GBF algorithm and the probability of the optimal path will be investigated in some special cases. Let P_E denote this difference given as follows:

$$\begin{aligned} P_E = \quad & P\left(\sum_{(u,v)\in R_{\text{opt}}} \delta_{(u,v)} \leq D\right) - \\ & P\left(\sum_{(u,v)\in R^*} \delta_{(u,v)} \leq D\right), \end{aligned} \tag{26}$$

which is referred to as "probability error". First, the expected value of (26) will be calculated for the following special example:

- Given a graph $G(V, E, C)$ where only two paths exist between the nodes $s, f \in V$: R_{opt} and R^*.
- $z \in V$ is a node contained by both R_{opt} and R^* in a way that both paths run together between s and z. Let R_c denote this common part (see Fig. 1.a). Furthermore, let R_a denote the part between z and f of R_{opt}, while R_b refers to the part between z and f of R^*. Then $R_{\text{opt}} = R_a \cup R_c$ and $R^* = R_b \cup R_c$.
- Assume that $C(R_b) \leq' C(R_a)$, while the inequality $\beta(C(R_b), C(R_c)) \leq' \beta(C(R_a), C(R_c))$ is not fulfilled, which implies that the GBF algorithm finds R^* instead of R_{opt}.
- Let L_a, L_b, L_c refer to the hop counts of the path R_a, R_b, R_c, respectively.
- Assume that grid τ is equidistant and the value $d_{(u,v)} \in \tau$ is advertised by link (u, v). Let d_a, d_b, d_c denote the sums of the advertised delay along the paths R_a, R_b, R_c, respectively; as $d_\alpha = \sum_{(u,v)\in R_\alpha} d_{(u,v)}\ \forall \alpha \in \{a, b, c\}$.
- Each link is supposed to have identical and known statistics given by the probability mass function $P(d_{(u,v)} = t_i)$. By the convolution, one can determine the probability of the delay of a path R according to the advertised thresholds: $\pi(d, |R|) = P\left(\sum_{(u,v)\in R} d_{(u,v)} = d\right)$.
- Define p_D as $p_D(R) = P\left(\sum_{(u,v)\in R} \delta_{(u,v)} > D\right)$ (probability of violating the QoS parameter) which only depends on the sum of the advertised delays and on the hop count, so the notation $p_D(d_\alpha, L_\alpha)$ can also refer to $p_D(R_\alpha)$ for all $\alpha \in \{a, b, c\}$.

Using the notations introduced above, the violation of monotonity criterion C4 can be written as follows:

$$\begin{aligned} & p_D(d_b, L_b) \leq p_D(d_a, L_a), \quad \text{but} \\ & \neg\left(p_D(d_b + d_c, L_b + L_c) \leq p_D(d_a + d_c, L_a + L_c)\right). \end{aligned}$$

In addition, error P_E is also obtained in the following fashion:

$$\begin{aligned} & P_E(d_a, d_b, d_c, L_a, L_b, L_c) = \\ & = \begin{cases} 0, & \text{if } p_D(d_b, L_b) > p_D(d_a, L_a) \\ \lceil p_D(d_b + d_c, L_b + L_c) - & \\ \quad p_D(d_a + d_c, L_a + L_c) \rceil^+ & \text{otherwise.} \end{cases} \end{aligned} \tag{27}$$

The expected value of P_E can be expressed according to the following theorem:

Theorem 3

$$\mathbf{E}\{P_E\} = \sum_{d_a \in \mathcal{D}_a} \sum_{d_b \in \mathcal{D}_b} \sum_{d_c \in \mathcal{D}_c} P_E(d_a, d_b, d_c, L_a, L_b, L_c) \cdot \pi(d_a, L_a)\pi(d_b, L_b)\pi(d_c, L_c), \quad (28)$$

where the sets $\mathcal{D}_a$, $\mathcal{D}_b$, $\mathcal{D}_c$ include the possible values of d_a, d_b, d_c, respectively (for all $\alpha \in \{a, b, c\}$ $\mathcal{D}_\alpha = \{L_\alpha t_1 + i\Delta : i = 0, 1, \dots L_\alpha(Z-1)\}$).

Proof:
P_E differs from zero only if the monotonity is not satisfied, so the GBF algorithm yields erroneous path. Since the three sums of (28) take all the possible delay combinations, all the possible values of P_E are taken into account. According to the definition of the expected value, the possible values of P_E -weighted with their probabilities- have to be added. Due to the link independence, the probability of each d_a, d_b, d_c combination can be expressed by the product of the probabilities $\pi(d_a, L_a), ..., \pi(d_c, L_c)$. From this, one can conclude that (28) exactly expresses the expected value of P_E. ■

Using this theorem, one can give an upperbound on $\mathbf{E}\{P_E\}$ in case of arbitrary graphs.

Theorem 4 *For all graphs $G(V, E, C)$, where $N = |V|$, the following inequality holds:*

$$\mathbf{E}\{P_E\} \leq \sum_{L_c=1}^{N-2} \sum_{L_a=1}^{N-1-L_c} \sum_{L_b=1}^{N-1-L_c} \sum_{d_a \in \mathcal{D}_a} \sum_{d_b \in \mathcal{D}_b} \sum_{d_c \in \mathcal{D}_c} \pi(d_a, L_a)\pi(d_b, L_b)\pi(d_c, L_c) \cdot P_E(d_a, d_b, d_c, L_a, L_b, L_c). \quad (29)$$

Proof:

In the first step, assume -as the worst case- that if there are two alternate paths between s and f which violate criterion C4, then it will affect the path selection, namely the selected path $R^* \neq R_{\text{opt}}$. One can note that the GBF algorithm begins to build a path towards both R_{opt} and R^* from node f. However, at their junction node (denoted by z in Fig. 1.a) R^* seems virtually better than R_{opt} (because C4 is not satisfied), so the solution by the algorithm cannot be R_{opt} any longer. Since z blocks the selection of R_{opt}, we will use the notation "blocking node" for z. Now assume that only one such "blocking node" arises while seeking R_{opt}. Namely, as Fig. 1.a depicts, $R_{\text{opt}} = R_a \cup R_c$ and $R^* = R_b \cup R_c$. Then, of course, the right hand side of (28) needs to be added for all the possible L_a, L_b, L_c hop counts, which leads to (29). This sum is an upperbound, because some of L_a, L_b, L_c combinations cannot occur in the graph.

In the second step, assume that there are more than one "blocking node" along R_{opt}. More precisely, the path $R_b \cup R_c$ is blocked by violating again C4. In this case, to get P_E, the difference $p_D(R_b \cup R_c) - p_D(R^*)$ has to be added

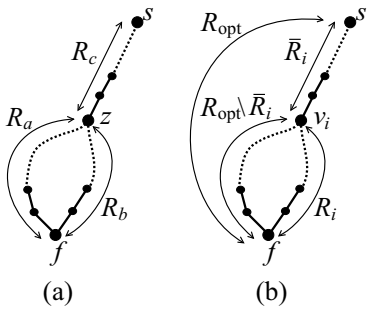


Figure 1: *Illustration of the monotonity violation in a simple example.*

to the error of $R_b \cup R_c$ (to $p_D(R_{\text{opt}}) - p_D(R_b \cup R_c)$). Now assume -as the worst case- that the GBF algorithm connects node f to each nodes of R_{opt} ($v_0, v_1, v_2, ..., v_{L-1}$; $v_0 = s$) with different paths R_i ($i = 0, 1, \dots L-1$; $R_0 = R^*$). Let $\overline{R}_i$ denote the part of R_{opt} from s to v_i. Then the expected value of the error caused by the "blocking node" v_i ($i = 1, 2, \dots L-1$) is upperbounded by the following expression:

$$\sum_{L_a=1}^{N-1-i} \sum_{L_b=1}^{N-1-i} \sum_{d_a \in \mathcal{D}_a} \sum_{d_b \in \mathcal{D}_b} \sum_{d_c \in \mathcal{D}_c} \pi(d_a, L_a)\pi(d_b, L_b) \cdot \pi(d_c, i) \cdot P_E(d_a, d_b, d_c, L_a, L_b, i), \quad (30)$$

where now index a refers to the properties of the subpath $R_{\text{opt}} \backslash \overline{R}_i$, b to the properties of R_i, while c to the properties of $\overline{R}_i$ (see Fig. 1.b), respectively. We stated that the errors caused by violations of C4 were aggregated, that is why (30) has to be added for all the possible $\overline{R}_i$ ($i = 0, 1, \dots L-1$) in order to estimate the overall error. Then (29) is obtained if we substitute $N-1$ (the upperbound of the hop count) in the place of L.

According to the derivation above, no matter how many C4 violations occur, the starting edge of R_{opt} will be the starting edge of R^*, too. Namely $(s, v_1) \in R_{\text{opt}} \cap R^*$. However, in practice this is not necessarily true, because it can occur that the error of the path built from v_1 is so high, that the node s selects a path avoiding v_1 so $(s, v_1) \notin R^*$. Due to the algorithm, the error of the obtained R^* will be lower than that of the path crossing v_1. Thus, the upperbound of (29) is still correct. This is the end of the proof. ■

Unfortunately, the evaluation of (29) needs $O(N^6 Z^3)$ steps, which seems very time-consuming. However, a more favourable complexity can be obtained by deriving a more exact bound. The number of steps is given by the following expression:

$$\sum_{L_c=1}^{N-2} \sum_{L_a=1}^{N-1-L_c} \sum_{L_b=1}^{N-1-L_c} \sum_{d_a \in \mathcal{D}_a} \sum_{d_b \in \mathcal{D}_b} \sum_{d_c \in \mathcal{D}_c} 1 \quad (31)$$

which can be upperbounded taking into account that $|\mathcal{D}_a| = L_a(Z-1) + 1 \leq L_a Z$ and similarly $|\mathcal{D}_b| \leq L_b Z$,

$|\mathcal{D}_c| \leq L_c Z$:

$$Z^3 \sum_{L_c=1}^{N-2} \sum_{L_a=1}^{N-1-L_c} \sum_{L_b=1}^{N-1-L_c} L_a L_b L_c = \\ = \frac{Z^3}{4} \sum_{L_c=1}^{N-2} L_c (L_c - N)^2 (L_c - N + 1)^2 \quad (32)$$

From this, the following term can be deduced:

$$\frac{Z^3 N}{120}(N-1)(N^4 - 2N^3 - 2N^2 + 3N + 2). \quad (33)$$

The table below shows this value for some graph sizes:

N	number of steps
10	$Z^3 \cdot 5,874$
20	$Z^3 \cdot 453,663$
30	$Z^3 \cdot 5,468,617$
40	$Z^3 \cdot 31,575,986$
50	$Z^3 \cdot 122,401,020$

where typically $z = |\tau| = 5 \ldots 10$. Fortunately, plenty of components in (29) are zero so the upperbound of the error based on (29) can be calculated in a finite time (e.g. in the case of even $N = 50$ and $Z = 50$, evaluating (29) lasts about two days with a simple Pentium). In the next section this bound and the average error based on simulations will be compared in case of a particular network.

7 Numerical results

The performance analysis of the presented method was carried out by using real link parameters and delay requirements on a modification of the well-known ANSNET topology (backbone of the USA). Firstly, the GBF algorithm was examined with respect to the "probability error". After this, the underlying efficiency that can be achieved by the GBF algorithm was demonstrated in comparison to the present routing algorithm using deterministic metric (similarly to the QOSPF standard).

7.1 The error of the GBF algorithm

In the course of the numerical examinations, the error given by (29) was evaluated as the function of the delay requirement D by using different graph sizes N, delay interval lengths Δ and distribution models (see Figures 2-4). The delay of each link is assumed to take its value uniformly from set $(0, 50ms)$, so the distribution $P(d_{(u,v)} = t_i)$ $(i = 1, 2, \ldots, Z)$ is uniform as well.

Figure 2 shows that the calculated upperbound of the expected value of the error (henceforth, in short: error) has a maximum. Since D was examined only between 30ms and 230ms, the initial parts of the curves run together. That is why, the curve of $N = 16$ already gives an appropriate approximation for arbitrarily big graph sizes N. (One can see that the curve of $N = 40$ hardly differs from the one of $N = 16$.) The analysis was performed assuming "on-off"

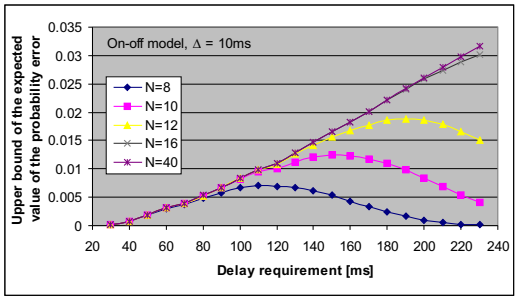


Figure 2: *The upperbound of* $\mathbf{E}\{P_E\}$ *as the function of QoS criterion* D *assuming "on-off" model. The curves are parametered by graph size* N. *(*$\Delta = 10ms$*)*

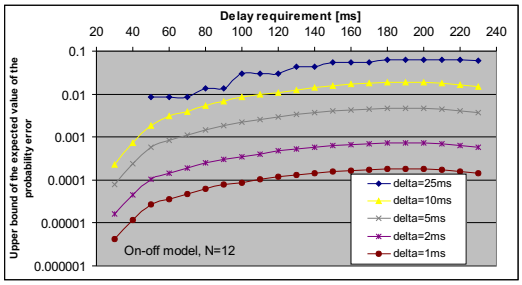


Figure 3: *The upperbound of* $\mathbf{E}\{P_E\}$ *as the function of* D*, parametered by interval length* Δ. *(*$N = 12$*)*

model and $\Delta = 10ms$ (equidistant grid).
One can observe in Figure 3 how the error depends on Δ as the function of D (assuming "on-off" model, $N = 12$). The bigger Δ is, the greater error the GBF algorithm has. Finally, Figure 4 depicts the error curves by using different distribution models (in the case of $N = 12$ and $\Delta = 10ms$). In sum, we can state that the error is below $n \cdot 0.01$ in the examined practical case. Namely, a very efficient method was obtained for approximating the original NP-hard MLP selection.

7.2 Evaluating the error on a real network

The performance of the GBF algorithm was investigated on a modification of the ANSNET backbone topology [15] with improved connectivity (see Fig. 5). (Its connectivity was improved for ensuring the existence of multiple paths.) For the sake of realistic simulations, link delays were generated uniformly from the set $(0, 50ms)$ and the delay requirement was chosen between 30ms and 160ms (or 230ms) [16]. Firstly, the difference between the probabilities of the path given by the GBF algorithm and that of the optimal path given by solving task (5) with exhaustive search was averaged for all the possible paths (between each starting node and each destination node) in case of 50 distinct delay realizations. (Let $\mathcal{G}$ denote these network situations as a graph set.) This average error was defined by

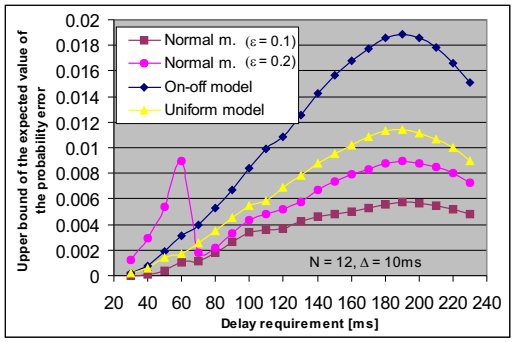


Figure 4: *The upperbound of* $\mathbf{E}\{P_E\}$ *as the function of* D *assuming different distribution models. (*$N = 12$, $\Delta = 10$*)*

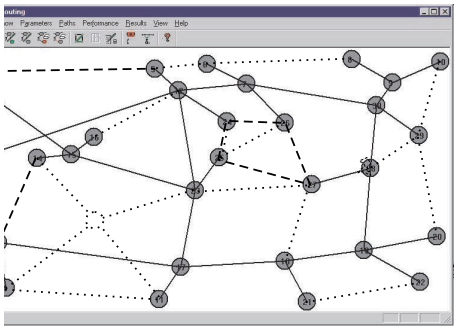


Figure 5: *The ANSNET topology with improved connectivity. (Dashed lines: modification of type I, dotted lines: modification of type II.)*

the following form:

$$\tilde{P}_E(D) = \frac{\sum_{G\in\mathcal{G}} \sum_{s\in V} \sum\limits_{\substack{f\in V \\ f\neq s}} \psi(G,D,s,f)}{|\mathcal{G}|\cdot|V|\cdot(|V|-1)} \tag{34}$$

where

$$\psi(G,D,s,f) = P\left(\sum_{(u,v)\in R^{s\to f}_{\mathrm{opt}}} \delta_{(u,v)} \le D\right) - P\left(\sum_{(u,v)\in R^{*\,s\to f}} \delta_{(u,v)} \le D\right). \tag{35}$$

Figure 6 depicts this average error as a function of the delay requirement D by using "on-off", normal ($\epsilon = 0.1$ and 0.2) and uniform model. (Equidistant grid was used over the delay scale: $\Delta = 10ms$). Here, the simulation was run for only the modification of type I of the ANSNET topology in order that the exhaustive search can be performed. As can be seen, the average was in the range of $n \cdot 0.001$, which means that (29) far overestimates the average error. (One must note that already (29) yields error level low enough.)

In what follows, the performance of the GBF algorithm will be analysed in comparison with routing with deterministic metric according to the advertised intervals. Since the exact delay information is unknown, for the sake of simplicity let the advertised lower interval limit correspond to

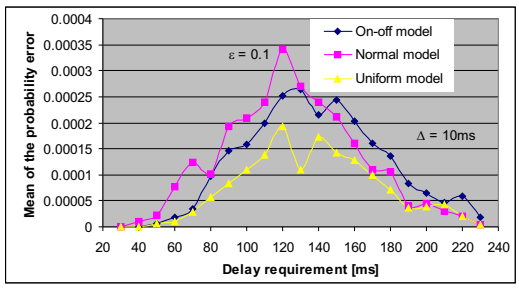


Figure 6: *The average error of the GBF algorithm as the function of QoS criterion* D *by using different distribution models. (*$\Delta = 10ms$*)*

the weights for the BF algorithm. Let $\kappa_{(u,v)}$ denote the actual delay of link (u,v) (current value of $\delta_{(u,v)}$) which is unknown in reality but known during the simulation. Furthermore, $d_{(u,v)} \in \tau$ refers to the current lower interval limit (i.e. if $d_{(u,v)} = t_i$, then $\delta_{(u,v)} \in (t_i, t_{i+1})$). Let $R^{s\to f}_{\mathrm{opt}}$ denote the shortest path between s and f according to the "unknown" $\kappa_{(u,v)}$ metric, and let $R^{s\to f}_{\mathrm{det}}$ refer to the shortest path according to the $d_{(u,v)}$ metric as follows:

$$\begin{aligned} R^{s\to f}_{\mathrm{opt}} &: \min_{R\in\mathcal{R}_{sf}} \textstyle\sum_{(u,v)\in R} \kappa_{(u,v)}, \\ R^{s\to f}_{\mathrm{det}} &: \min_{R\in\mathcal{R}_{sf}} \textstyle\sum_{(u,v)\in R} d_{(u,v)}. \end{aligned} \tag{36}$$

Let us introduce an indicator function $\iota(R)$ which gives value 1 if R meets the QoS criterion, namely:

$$\iota(R) = \begin{cases} 1 & \text{if } \sum_{(u,v)\in R} \kappa_{(u,v)} < D \\ 0 & \text{otherwise.} \end{cases} \tag{37}$$

If $\iota(R) = 1$, then path R is called "good", otherwise "bad". Figures 7 and 8 exhibit what proportion of the routes by the GBF algorithm is "good", when the deterministic routing by the lower limits cannot find "good" path, provided "good" path exists. This performance measure is formulated by the following expression:

$$\frac{\sum\limits_{G\in\mathcal{G}} \sum\limits_{s\in V} \sum\limits_{f\in V\setminus\{s\}} \iota(R^{s\to f}_{\mathrm{GBF}}) \cdot (1 - \iota(R^{s\to f}_{\mathrm{det}}))}{\sum\limits_{G\in\mathcal{G}} \sum\limits_{s\in V} \sum\limits_{f\in V\setminus\{s\}} (1 - \iota(R^{s\to f}_{\mathrm{det}})) \cdot \iota(R^{s\to f}_{\mathrm{opt}})} \tag{38}$$

The analysis was carried out assuming the three delay distribution models in case of several numbers of (equidistant) delay intervals and different delay requirements. Modification of type II of the ANSNET topology was used, where 6000 delay realizations were generated. As can be seen, in 70-80% of the cases when R_{det} was "bad", the GBF algorithm already found a "good" path.

Finally, it was investigated in what rate the deterministic routing (where link metrics are defined by the lower delay interval limits) and the GBF algorithm found a "good" path (provided "good" path existed), in the case of 6000 delay realizations of the modified ANSNET (type II). The examined performance measure is defined more precisely by the

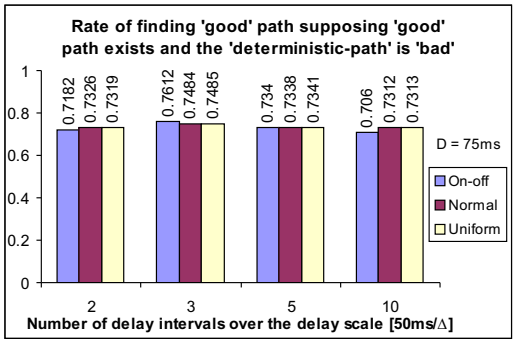


Figure 7: *Rate of finding "good" path supposing it exists when the deterministic routing does not find "good" path (under $D = 100ms$ and 6000 ANSNET (type II) delay realizations).*

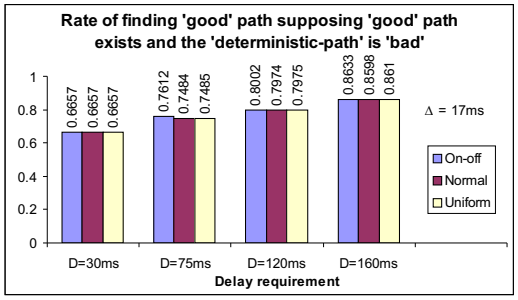


Figure 8: *Rate of finding "good" paths under the conditions of Fig. 7 as the function of delay requirement D. ($\Delta = 17ms$)*

following term:

$$\frac{\sum_{G\in\mathcal{G}}\sum_{s\in V}\sum_{f\in V\setminus\{s\}}\iota(R^{s\to f}_{\text{by the given algorithm}})}{\sum_{G\in\mathcal{G}}\sum_{s\in V}\sum_{f\in V\setminus\{s\}}\iota(R^{s\to f}_{\text{opt}})}. \qquad (39)$$

Since, according to the discussion above, essential difference was not found among the distinct models, only the normal model was examined here by using different distributions of the delay thresholds in τ. Figure 9 summarizes the results as the function of the number of delay intervals. Column-pair 3^* was taken assuming non-equidistant grid: $\tau = \{0, 10ms, 26ms\}$. This indicates that routing really yields better performance in the case of a non-equidistant grid.

As Figure 9 shows, if the inaccuracy of link delays is taken into account by the GBF algorithm, QoS routing far outperforms the deterministic routing (where link metrics are defined by the lower limits). The less delay intervals are used, the greater difference of performance occurs between the two methods which emphasizes the superiority of the new method.

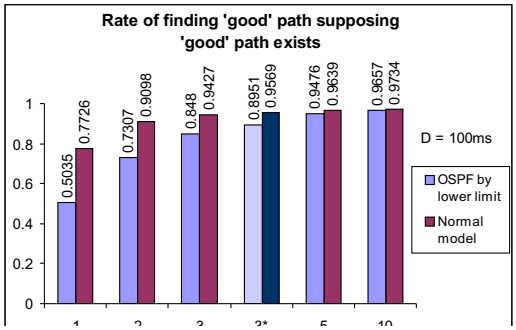


Figure 9: *Rate of finding paths fulfilling the QoS criterion, provided such path exists, taking into account 6000 delay realizations of the modification of type II of the ANSNET. ($D = 100ms$)*

8 Conclusion

In the paper a novel method was proposed to carry out QoS routing with additive (delay-type) link metrics. The accurate delay values were not available, only the probability distribution of link-delays were assumed to be known. The task was transformed -with abstract algebraic tools- into a form which can be solved by the Bellman-Ford algorithm. Unfortunately, the solution has an approximative nature, but providing an upperbound on the approximation error, the performance of the method lends itself to calculation. The developed algorithm can very efficiently perform QoS routing with inaccurate information. This statement is also supported by extensive simulations. The real benefit of the proposed method surfaces most strikingly in comparison with deterministic routing (where link metrics are defined by the lower interval limits and merely some intervals cover the delay scale). Unfortunately, in this case the uniformly distributed link-delay hypothesis is not valid any more. This is why, further models - which describe the properties of links more precisely - need to be investigated.

Acknowledgement

The author is grateful for the comments and guidelines he received from Dr János Levendovszky during his research and from Mónika Micsonai in the course of writing the paper.

Appendix

The proof of Theorem 1:
Firstly, we show that the label of at least one node takes its final value after each step so the routing procedure surely terminates after $N-1$ steps. Let $S[k-1]$ denote the set of nodes the labels of which are stabilized at the $(k-1)$th step, namely: $S[k-1] = \{w | Q_v[k-1] = Q_v[k+i], i \geq 0\}$. Let

$U[k-1]$ refer to the set of the neighbours of the elements in $S[k-1]$ that are outside $S[k-1]$, more precisely:

$$U[k-1] = \{u|(u,v) \in E, u \notin S[k-1], v \in S[k-1]\}\,.$$

Define two subsets in $U[k-1]$ as:

$$U'_s[k-1] = \{u|u \in U[k-1], z_u[k] \in S[k-1]\}$$
$$U'_{\min}[k-1] = \left\{u|d_{uf}[k] \sim \min^*_{w\in U[k-1]} d_{wf}[k]\right\}.$$

In the first step, we claim that $U'_s[k-1] \cap U'_{\min}[k-1] \neq \emptyset$. Indeed, assume that there is not an $u \in U'_s[k-1]$ (i.e. $z_u[k] \in S[k-1]$) that $d_{uf}[k] \sim \min^*_{w\in U[k-1]} d_{wf}[k]$ could hold. That is to say,

$$\min^*_{w\in U[k-1]} d_{wf}[k] <^* d_{vf}[k] \tag{40}$$

for all $v \in U'_s[k-1]$. Furthermore, the indirect assumption also claims that $z_u[k] \in U'_s[k-1]$ for all $u \in U'_{\min}[k-1]$ (i.e. the $d_{uf}[k]$ is improved from an element in U'_s), namely:

$$\beta\left(d_{vf}[k-1], c_{vu}\right) <^* d_{uf}[k-1]$$
$$\longrightarrow \quad d_{uf}[k] = \beta\left(d_{vf}[k-1], c_{vu}\right),$$

where $v \in U'_s[k-1]$. According to the algorithm, the inequality $d_{vf}[k] \leq^* d_{vf}[k-1]$ is satisfied, and from the non-negativity: $d_{vf}[k-1] \leq^* \beta\left(d_{vf}[k-1], c_{vu}\right)$ holds. Thus, taking into account the transitivity, the inequality $d_{vf}[k] \leq^* d_{uf}[k]$ is obtained, which contradicts (40). Therefore, $U'_s[k-1] \cap U'_{\min}[k-1] \neq \emptyset$ (independently of k).

Now assume that $u \in U'_s[k-1] \cap U'_{\min}[k-1]$. We will demonstrate that the label (Q_u) of this u will not change in later steps. Assume the contrary: there is an index i and a node $w \in T_u$ that $\beta\left(d_{wf}[k+i], c_{uw}\right) <^* d_{uf}[k]$. Taking into account the non-negativity and the transitivity, this would correspond to the satisfaction of

$$d_{wf}[k+i] <^* d_{uf}[k]. \tag{41}$$

On the one hand, this statement could be true if there was an edge improving d_{wf} from $S[k-1]$. However, in that case, inequality $d_{wf}[k] <^* d_{uf}[k]$ would also hold. Because of the assumption $u \in U'_{\min}[k-1]$ the previous inequality cannot be satisfied. On the other hand, because $d_{uf}[k] \leq^* d_{vf}[k]$ for all $v \in V\backslash S[k-1]$ at the kth step (seeing that $u \in U'_{\min}[k-1]$), there is not a node outside $S[k-1]$ from which d_{wf} could be improved (which could induce the situation (41)). Therefore, using the non-negativity, d_{wf} could be improved to "higher level" than $d_{uf}[k]$ is, namely: $d_{uf}[k] \leq^* d_{wf}[k+i]$ for all $i > 0$. This contradicts (41), i.e., label Q_u of an element $u \in U'_s \cap U'_{\min}$ is stabilized ($z_u[k] \in S[k-1]$ and $d_{uf}[k] = d_{uf}[k+i]$), which implies that $S[k] := S[k-1] \cup \{u\}$. Since the line of reasoning given above is independent of k, the algorithm terminates at the $(N-1)$th step and $S[N-1] = V$.

Regarding the running time, each node controls the possibility of improving its label for all the neighbouring nodes at each step, which demands $2|E| \cdot (|V|-1)$ evaluations of $\leq^*$ and β.

As each stabilized node-label was updated from an earlier stabilized node-label (seeing that parent node $z_u[k] \in S[k-1]$, $u \in U'_s[k-1]$), at the end of the algorithm the edge set $\{(u, z_u) \in E|u \neq z_u, \forall u \in V\}$ defines a spanning tree starting from f, provided that $G(V,E,C)$ is connected. Thus, the paths defined by parent nodes z_v are connected. Now, we will prove that these paths are optimal if the criterion of monotonity C4 is fulfilled. We will argue by induction.

For k=1: d_{uf} of the element $\{u\} = S[1]\backslash\{f\}$ is evidently optimal, because $d_{uf} = c_{uf} = \min^*_{v\in T_f} c_{vf}$ according to the algorithm. Now assume that d_{vf} is minimal for all $v \in S[k-1]$. Then it has to be demonstrated that the d_{uf} of the element $\{u\} = S[k]\backslash S[k-1]$ is optimal, too. Hereafter, let z denote the parent node (z_u) of this u. From the algorithm we know that

$$\beta\left(d_{zf}, c_{uz}\right) \leq^* \beta\left(d_{wf}, c_{uw}\right), \tag{42}$$

where w is another neighbour of u: $w \in T_u\backslash\{z\}$. At the same time, suppose that there exists a path $\tilde{R}_{zf} \neq R^*_{zf}$ (where R^*_{zf} refers to the path built by the algorithm) that $d_{zf} \leq^* C\left(\tilde{R}_{zf}\right)$, while $\beta\left(C\left(\tilde{R}_{zf}\right), c_{uz}\right) <^* \beta\left(d_{zf}, c_{uz}\right)$. However, this contradicts criterion C4, so there is no other path via z from u to f whose "cost-level" could be "below" d_{uf}. Now assume that there exists another $w \in T_u\backslash\{z\}$ via which a better path ($\tilde{R}_{wf}$) is obtained from u to f than via z (R^*_{wf}). More precisely,

$$\exists \tilde{R}_{wf} \neq R^*_{wf}: \; d_{wf} \leq^* C\left(\tilde{R}_{wf}\right),$$
$$\text{while } \beta\left(C\left(\tilde{R}_{wf}\right), c_{uw}\right) <^* \beta\left(d_{zf}, c_{uz}\right).$$

From this, using (42) and the transitivity, we get the inequality $\beta\left(C\left(\tilde{R}_{wf}\right), c_{uw}\right) <^* \beta\left(d_{wf}, c_{uw}\right)$, which could just result in violating C4. That is why one can claim that the overall "cost-level" d_{uf} of the path R^*_{uf} given by the algorithm is minimal, too. This completes the proof. ■

References

[1] Lorenz, D., Orda, A.: "QoS routing in networks with uncertain information", *IEEE/ACM Trans. Networking*, vol. 6., December, 1998.

[2] Apostolopoulos, Guérin, Kamat, Tripathi: "Quality of Service Based Routing: A Performance Perspective", *Proceedings of SIGCOM*, pp. 17-28, Vancouver, Ontario, Canada, September 1998.

[3] Jaffe, J.: "Algorithms for Finding Paths with Multiple Constraints", *Networks*, Vol 14, pp 95-116, 1984

[4] Guérin, R., Orda, A.: "QoS routing in networks with inaccurate information: theory and algorithms", IEEE/ACM Trans. Networking, vol. 7., June, 1999.

[5] Levendovszky J., Rétvári G., Dávid T., Fancsali A., Vegso Cs.: "QoS routing in packet switched netwoks - novel algorithms for routing with incomplete information", *9th IFIP Conference on Performance Modelling and Evaluation of ATM & IP Networks*, 2001, Budapest

[6] PNNI Specification Working Group: "Private Network-Network Interface Specification Version 1.0", *ATM Forum*, March 1996.

[7] Guérin, R., Kamat, S., Orda, A., Przygienda T., Williams, D.: "QoS Routing Mechanisms and OSPF Extensions" *RFC 2676*, IBM, Technion, Lucent, December 1998.

[8] Shaikh, A., Rexford, J., Shin, K.: "Dynamics of Quality-of-Service Routing with Inaccurate Link-State Information", *Technical Report CSE-TR-350-97*, Computer Science and Engineering Division, Dept. of Electrical Engineering and Computer Science, University of Michigan, November 1997

[9] Garey, M.R. and Johnson, D.S., *Computers and Intractability*, Freeman, San Francisco, 1979.

[10] Levendovszky J., Vegso Cs.: "QoS routing with incomplete information using Large Deviation Approach", *2001 Polish-Czech-Hungarian Workshop on Circuit Theory, Signal Processing, and Telecommunication Networks*, Budapest, 2001.

[11] Cormen, T. H., Leiseron, C. E., Rivest, R. L., *Introduction to Algorithms*, Cambridge, MA: MIT Press, 1990.

[12] Birkhoff, G. and Bartee, T. C., *Modern Applied Algebra*, New York, NY: McGraw-Hill, 1970.

[13] Prékopa A., *Valószinüségelmélet müszaki alkalmazásokkal* (Probability theory with technical applications), Müszaki Könyvkiadó, Budapest, 1962. (in Hungarian)

[14] Levendovszky J., Imre S., Pap L., E.C. van der Meulen, Varga B., *Comparative Analysis of Call Admission Control Algorithms for ATM Networks*, Annual Scientific Progress Report, COPERNICUS C579, August, 1995.

[15] Comer, D. E., *Internetworking with TCPIP*, Volume I., Prentice Hall, 1995.

[16] Chen, S., Nahrstedt, K.: "Distributed QoS Routing with Imprecise State Information", *Proceedings of 7th IEEE International Conference on Computer, Communications and Networks*, Lafayette, LA, pp. 614-621, October 1998.

Conscious Informational Entities

Anton P. Železnikar
Volaričeva ul. 8, Ljubljana, SI–1111
s51em@hamradio.si

Keywords: attention, cognitive-emotional paradigm, complexity, emergentism, entity name, entropon, entroponic, entroponizing, informon, informonic, informonizing, intention, metaphysicalism, phenomenalism, the metatheory of the conscious, the philosophy of the informational

Received: July 22, 2003

This paper deals with the constitution and organization of a conscious entity. Informon, as an ordered and named conscious entity, informs in its equally named and informationally disordered (chaotic, irregular) environment, called entropon. Both informon and entropon overlap informationally more or less with other informons and entropons constituting a local or global informational space.

A conscious system, as known from the naturalistic point of view, is necessarily a cognitive-emotional system together with other components concerning sensuality, attention, motivation. homeostasis, behavior, motorics, etc., all of them possessing a specific intention (orientation, goal-directedness, decision-making specificity) within the conscious informing. Concrete cognitive, emotional, and otherwise meaningly organized entities come into the conscious foreground.

Artificial conscious systems can differ substantially from the natural ones in the functional domain, regarding the future complexity, memory, global interconnection, and informational search possibilities. Artificial entropons can serve as absolute memories for experiences (components, parts of components) and, in this way, can enable informons to use the entire precisely, regularly, and disorderly structured previous experience and their lumps.

1 Intercomplex and interconscious organization of informational entities

Components of a conscious system are informationally fused in nature. In addition, a circular metaview of causation comes fore, in which stimulus and response take the role to be both cause and effect. Rotation of operands in a circular formula scheme is the consequence of this fact [19, 20, 21]. Circular organization of informational entities reflects in informational metaphysicalism [21], being the condition sine qua non in the organization of conscious components (see, for instance, R. S. Lazarus [15]).

Complexity [4] and metaphysicalism [19, 20], determining conscious entities, can be explained through an enormous degree of informational interrelationship. In a conscious system, conscious entities (components) are

> *interactive, interassociative, intercausal, intercircular, interconnected, interconscious, intercommunicational, interdependent, interdistributed, intereffective, interemergent, interferential, interintentional, interreflective, interrelational, interspontaneous, interweaved*, that is, *interinforming* and *interinformational.*

This kinds of interselfness constitute the intercomplexness of entities, that is, the distributed complexity of entities and their (entroponic) environments, in which each of entities can inform consciously.

The consequence of these properties is that a conscious system is organized intercomponetially, or naturally, artificially, pragmatically, and spontaneously, in an intercognitive, interemotional, intermotivational, interattentional, interbehavioral, interhomeostatical emergent way, and the like. Circularity in the form of metaphysicalism remains the main ability of the conscious informational phenomenon. Informational betweenness or interness (interism) is the consequence of connectedness and distributiveness of conscious components fused informationally in the system. In this context, metaphysicalism is the metaphor for a basic organization of conscious entities, for instance, in the form of informons in informonic environments [19, 20], and in this way being cognitive, emotional, motivational, attentional, homeostatic, behavioral, etc.

Within conscious systems, informational complexness of components reaches the extreme forms in number and interism, causing the possibility for emergence of informons, coming into the conscious attention. In such an environment, any component emerged, can immediately appropriate to it corresponding conscious property or concrete functional organization, for it has the access to the subconsciously complex domain of informational lumps in which conscious abilities are memorized, associatively conditioned (interrelated), and being experientially on disposal.

Artificial conscious systems can hardly ignore the so-called cognitive-emotional paradigm, constituting the con-

cept of consciousness as a natural system of awareness, self-awareness, and behavior of living creatures. The artificial aspires to imitate the natural consciousness with the aim, to make artificial conscious systems adopted to the human culture and the way of understanding. In this respect, cogniton-driven and emotion-driven robots are sine qua non in the field of sociable robots, designed to be spiritual friends and practical helpers in the environment of human everyday life.

2 Complexity and metaphysicalism as general principles of conscious informing

Complexity in number, structure, and organization of informational components is one of the conditions necessary for the emerging of conscious systems [4, 19, 20]. Human brain is an example of the large number of neurons, synapses and other ingredients connected perplexedly in an informational way, in which the conscious mind can emerge and develop as the complexity of thinking, its informational coherence, and decoherence (see decoherence in quantum computing in Kurzweil [14], pp. 110–118 and informational gestalt decoherence in Železnikar [21], Subsect. 8.9). Thus, the mind for itself becomes the complexity, in which mind can emerge and become more complex. That what in mind is accumulated are individual experiences, a sort of conscious and subconscious result of success and failure, within an intention of the individual living or artificial conscious system.

Metaphysicalism is a metaphor for a specific organization of informing of entities in the domain of consciousness [19, 20, 21]. It concerns intentional informing, intentional counterinforming and intentional informational embedding in a complex circular informational organization. Metaphysicalism always concerns an object, a certain informational entity, to which it is intentionally directed. We always speak about entity's metaphysicalism, with a specific sort of meaning rooting in entity's intention, that is, in its simple or complex name, that is, name's meaning, being a complex formula system. Name performs as an impulse for the emergence of the complex organization, preserving and generating the meaning of the name, as it happens in the case of informon, developing in the emerging informational space.

3 Entropon and informon

Entropon $\overline{\boldsymbol{\alpha}}$ is the α-named informational background of informon $\underline{\boldsymbol{\alpha}}$, its subconscious or unconscious domain, from which the informon emerges as a conscious component at the very moment. Entropon is the entropic, informationally disordered state within informational space, concerning a named informational entity, including both well-formed and irregular informational items, that is, formulas and formula systems, as well as parts of the both.

At the very beginning, the informon, as an instantaneous event of experience, emerges through its name or name phrase α out of informon's entropic (entroponic) background, fitting informationally each new situation in the ongoing moment. For example, a flow of consciousness in people's speech means the occurrence of events as informons, which follow one another spontaneously, unpredictably in details, and depending on inner (conscious, subconscious, intentional) and outer (sensory) impulses, events, states of experience, circumstances, situations, emotions, attitudes, associations, perceptions, cognition, attentions, motivations, etc.

Entropon is an informational potential of informon[1]. It is an environment, from which an informon emerges, being structured consciously by decomposition, that is, shaped regularly in an informational way. In this respect, as an informational component, informon has its fore-having, foresight, and fore-conception (Heidegger [11] p. 195)[2], just in informon's decomposition in the area of its entropon. In this view, entropon is the available unordered informational domain for the informational (informonic) decomposition out of the disorder of entropon, into the reasonable order of informon.

In the way informons overlap each other, their entropons overlap too in the informational superspace. Both constitute the so-called informational substance of consciousness, being modulated intentionally. In them, the individual consciousness is captured, possessing specific intentions. The quality of consciousness roots in the complexity of entroponic diversity and extensiveness, and in the rational and intuitive orderliness of informonic intentionalities. Reflexive consciousness is just an informonic intervening into the entropic history, is a circular and sufficiently complex memorizing and understanding of the happening of the present and the past, with the view to the future and, in this respect, is a circular decomposition of informons within occurring situations and moments. Consciousness is an informonic happening out of entroponic background—informational disorder of subconsciousness or the individually unconscious background.

The pair of informon and entropon, $(\underline{\boldsymbol{\alpha}}; \overline{\boldsymbol{\alpha}})$, is a correlative and informationally coherent system. Entropon $\overline{\boldsymbol{\alpha}}$ is the broadest non-erderly emerging complex informational environment, background, or domain being named by α. Operand α represents a simple operand (a simple word or word phrase name), a formula (sentence), or a formula system (a group of meaningly interrelated sentences). Entropon consists of informational lumps concerning the name α. In a moment, the name α comes to the conscious foreground, informon $\underline{\boldsymbol{\alpha}}$ begins to emerge out of the entropon $\overline{\boldsymbol{\alpha}}$. In these circumstances, informon as informational entity with metaphysicalistic organization, becomes a part, in

[1] Entropons can be compared with the contents of a quantum-understood black hole from which new matter can come to existence.

[2] In German (Heidegger [12] p. 153), these words are *Vorhabe*, *Vorsicht* and *Vorgriff*.

fact, an element of entropon $\overline{\boldsymbol{\alpha}}$. Thus, according to the topological organization of the informational [18], where a system of formulas is equivalent to a set of meaningly (informationally, intentionally) interrelated elements,

$$\underline{\boldsymbol{\alpha}} \in \overline{\boldsymbol{\alpha}}$$

This relation can hold permanently only for artificial conscious systems, while in natural conscious systems, the emerging informon $\underline{\boldsymbol{\alpha}}$ is liable to a gradual, time dependent decay.

In a natural conscious system, after the informon's actualization as a conscious experience, the absolute organization of informon $\underline{\boldsymbol{\alpha}}$, entering into its entropon $\overline{\boldsymbol{\alpha}}$, disintegrates gradually, according with the function of the working memory, into substantial, intentionally relevant parts, that is, becomes informational lumps straying in the subconscious substrate of entropon. This means that in the whole of the informon, some common operands disappear and the connection to the corresponding informons is disrupted, so it must be restored anew, when the informon as such is coming into the conscious foreground. On contrary, in an artificial conscious system, the absolute organization of informon $\underline{\boldsymbol{\alpha}}$ can remain preserved within entropon $\overline{\boldsymbol{\alpha}}$.

To the basic relation $\underline{\boldsymbol{\alpha}} \in \overline{\boldsymbol{\alpha}}$, the following formulas can be introduced:

$$\begin{array}{l}(\underline{\boldsymbol{\alpha}};\, \overline{\boldsymbol{\alpha}}) \Longrightarrow (\underline{\boldsymbol{\alpha}} \models \overline{\boldsymbol{\alpha}};\, \overline{\boldsymbol{\alpha}} \models \underline{\boldsymbol{\alpha}});\\ \overline{\boldsymbol{\alpha}} \rightleftharpoons (\underline{\boldsymbol{\alpha}};\, \overline{\boldsymbol{\alpha}});\\ \underline{\boldsymbol{\alpha}},\, \overline{\boldsymbol{\alpha}} \in \overline{\boldsymbol{\alpha}};\\ \left(\left[\,\right] \models \underline{\boldsymbol{\alpha}}\right) \Longrightarrow (\overline{\boldsymbol{\alpha}} \models \underline{\boldsymbol{\alpha}})\end{array}$$

The first formula is an implication considering the circular organization (the right side of $\Longrightarrow$) of the informon-entropon pair. By the second formula, a new denotation is introduced, called *informational space* of the name denotation α, that is, $\overline{\boldsymbol{\alpha}}$. Such a structure is reasonable because the informonic part $\underline{\boldsymbol{\alpha}}$ can be memorized on a storage medium from where the artificial conscious system can observe it or pick it up as an original informonic whole. In the complex structure of entropon $\overline{\boldsymbol{\alpha}}$, informon emerges into the conscious entity, having on informational disposal all other entropons, connected informationally by common operands occurring in existing entropons. Here, the so-called interism, discussed in Sect. 1, functions throughout the conscious system of entropons.

The third formula states that an informon $\underline{\boldsymbol{\alpha}}$ and its entropon $\overline{\boldsymbol{\alpha}}$ are members of informational space $\overline{\boldsymbol{\alpha}}$. The fourth formula of the array, $\left(\left[\,\right] \models \underline{\boldsymbol{\alpha}}\right) \Longrightarrow (\overline{\boldsymbol{\alpha}} \models \underline{\boldsymbol{\alpha}})$, brings the so-called informational nothingness (the unknown, marked by $\left[\,\right]$) into the discourse. Usually, informational nothingness is the metaphor for anything possible and $\left[\,\right] \models \underline{\boldsymbol{\alpha}}$ means that by informing of $\left[\,\right]$, $\underline{\boldsymbol{\alpha}}$ can come into existence. However, such a presumption implies (operator $\Longrightarrow$) that, in fact, informon $\underline{\boldsymbol{\alpha}}$ emerges out of entropon $\overline{\boldsymbol{\alpha}}$.

In a natural conscious system, we clearly experience that informon $\underline{\boldsymbol{\alpha}}$, if not stored on a medium, disintegrates after the emerging within a conscious state, that is,

$$\underline{\boldsymbol{\alpha}} \models_{\text{disintegrate\_within}} \overline{\boldsymbol{\alpha}}$$

Let α_j be a synonym (an alternative definitions) of α, where $\alpha \models_{\text{synonymously}} \alpha_1, \alpha_2, \ldots, \alpha_j, \ldots$ and $\alpha_1, \alpha_2, \ldots, \alpha_j, \ldots \models_{\text{synonymously}} \alpha$. Entropon $\overline{\boldsymbol{\alpha}}$ can be structured as

$$\underline{\boldsymbol{\alpha}}, \underline{\boldsymbol{\alpha_1}}, \underline{\boldsymbol{\alpha_2}}, \ldots, \underline{\boldsymbol{\alpha_j}}, \ldots \in \overline{\boldsymbol{\alpha}}$$

From the name α, synonymous names $\alpha_1, \alpha_2, \ldots, \alpha_j, \ldots$ come in, which can be understood as alternatives of α and, vice versa, alternative names impact informationally the initial name α. This interplay can be understood as a consequence of synonymous emerging of informons $\underline{\boldsymbol{\alpha}}, \underline{\boldsymbol{\alpha_1}}, \underline{\boldsymbol{\alpha_2}}, \ldots, \underline{\boldsymbol{\alpha_j}}, \ldots$ and their entroponic counterparts $\overline{\boldsymbol{\alpha}}, \overline{\boldsymbol{\alpha_1}}, \overline{\boldsymbol{\alpha_2}}, \ldots, \overline{\boldsymbol{\alpha_j}}, \ldots$. In an informational space $\overline{\boldsymbol{\alpha}}$ and $\overline{\boldsymbol{\alpha_j}}$, names of informons are not all for ever meaningly stable entities and, in a conscious environment, they can be informationally fine-tuned, specifically (culturally) modulated and, lastly, meaningly modified through the interinforming of informons and entropons.

In natural systems, entropons $\overline{\boldsymbol{\alpha}}$ and $\overline{\boldsymbol{\alpha_j}}$ become the history of decayed informons $\underline{\boldsymbol{\alpha}}$ and $\underline{\boldsymbol{\alpha_j}}$, where informational lumps (parts 'wandering' around) concerning former informons are gathered. In general, for informons $\underline{\boldsymbol{\alpha}}$ and $\underline{\boldsymbol{\alpha_j}}$,

$$\begin{array}{l}\underline{\boldsymbol{\alpha}} \models_{\text{disintegrate\_in}} \mathfrak{p}_{\text{part},1}\lceil\underline{\boldsymbol{\alpha}}\rceil, \ldots, \mathfrak{p}_{\text{part},i}\lceil\underline{\boldsymbol{\alpha}}\rceil, \ldots;\\ \underline{\boldsymbol{\alpha_j}} \models_{\text{disintegrate\_in}} \mathfrak{p}_{\text{part},j1}\left\lceil\underline{\boldsymbol{\alpha_j}}\right\rceil, \ldots, \mathfrak{p}_{\text{part},jk_j}\left\lceil\underline{\boldsymbol{\alpha_j}}\right\rceil, \ldots\end{array}$$

Informon $\underline{\boldsymbol{\alpha}}$ is a complex system of formulas φ_i and the corresponding subformulas are $\psi_{ij_i} \Subset \varphi_i$. Thus, for ψ_{ij_i}, as well-formed parts of φ_i, $i = 1, 2, \ldots, \mathbb{L}_{\varphi_i}$,

$$\psi_{ij_i} \in \overline{\boldsymbol{\alpha}}, \text{ where } \psi_{ij_i} \Subset \varphi_i \text{ and } \varphi_i \in \underline{\boldsymbol{\alpha}}.$$

$\mathbb{L}_{\varphi_i}$ is the number of all subformulas ψ_{ij_i} in formula φ_i [21]. Evidently, concrete informons as parenthesized formula systems do not enter in a natural entropon. They cannot be memorized precisely because of their complexity in a human conscious system. As we believe, their memorizing is approximative and superficial along some key words and phrases. Even more, the decay of informon formulas in an entropon can progress to a total destruction and, thus, the basic constituents of formulas like differently subscribed, however, to the informon adequate operands and operators, enter in the entropon. In this way, in entropons $\overline{\boldsymbol{\alpha}}$ and $\overline{\boldsymbol{\alpha_j}}$, there are meaningly suitable operators in regard to the meaning structures of informons $\underline{\boldsymbol{\alpha}}$ and $\underline{\boldsymbol{\alpha_j}}$, respectively, that is,

$$\models_{\text{particularized},m_{j_i}} \in \overline{\boldsymbol{\alpha_j}}, \overline{\boldsymbol{\alpha}};\ i, m_{j_i} = 0, 1, 2, \ldots$$

where $\alpha \rightleftharpoons \alpha_0$. So, entropon $\overline{\boldsymbol{\alpha}}$ keeps a large number of constituents concerning different informons $\underline{\boldsymbol{\alpha_j}}$ named α_j being deduced synonymously from the initial intention or name α_0, giving rise to the emergence of the informons particular intentions.

In an artificial conscious environment, informon $\underline{\boldsymbol{\alpha}}$ can remain preserved in entropon $\overline{\boldsymbol{\alpha}}$ as a specific formula system, as it has emerged during its conscious cycle (e.g. being written, stored on a medium). In general, it has the

system form

$$\underline{\boldsymbol{\alpha}} \rightleftharpoons \begin{pmatrix} \varphi_1; \\ \varphi_2; \\ \vdots \\ \varphi_i; \\ \vdots \end{pmatrix}, \quad \text{where,} \quad \underline{\boldsymbol{\alpha}} \in \overline{\boldsymbol{\alpha}}$$

and is connected with other informons and entropons via the common operands $\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{ij_i}, \ldots, \alpha_{in_{\varphi_i}}$ of formulas $\varphi_i \lfloor \alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{ij_i}, \ldots, \alpha_{in_{\varphi_i}} \rfloor$.

4 Defining informon and entropon philosophically and formally

The definition of a complex and metaphysically organized informational entity—conscious informational entity—and its direct environment is important for understanding and formal implementation of both informon and entropon. The entity definition means that a concrete implementation may follow in an individual mind and computer net environment. In this context we are speaking specifically and concisely in a formal way about *informational informon* and *informational entropon*, distinguishing them from other concepts using the same basic names, for instance, in quantum mechanics (quantum informon and quantum entropon)[3]. While the quantum informon and entropon are defined as simple information carrying particles, informational informon and entropon are named as the most complex distributed formula systems within a conscious system.

The first published definition of the (informational) informon was in German, obtained from [21] by improvement in verbal and formal sense. According to the definition in German [19][4],

> *Informon is a new concept for the complexity of a conscious entity. Each informon as an informational entity is provided with its own consciousness. An individual consciousness consists of informons. An informon is in a given consciousness distributed informationally and, in this way, more or less, existent in other informons. Consciousness is nothing other than an informational overlapping of informons, and being an informon by itself.*

The artificial informon, named α, can be defined precisely, for instance, as

$$\underline{\boldsymbol{\alpha}} \rightleftharpoons_{\text{def}} \underline{\mathfrak{M}_{\triangleright}^{\circlearrowright\parallel}\lceil \boldsymbol{\alpha} \rceil}, \text{ where } \triangleright \in \{\rightarrow, \leftarrow, \rightleftarrows, (\rightarrow, \leftarrow)\}$$

$\mathfrak{M}_{\triangleright}^{\circlearrowright\parallel}\lceil \alpha \rceil$, as a pre-informonic entity, denotes the metaphysicalistic decomposition of name α, according to the subscript $\triangleright$. It delivers the basic metaphysicalistic organization to α (intentional, counter-intentional, and embedding-intentional informing), from which through complexity, that is, complex connectivism in the conscious system, informon as a conscious component can emerge. Informon's connectivism concerns all available operands in the system, concerning informon's name directly and indirectly, associatively, and otherwise possibly. This situation can be captured graphically in Fig. 1. Probably, informon $\underline{\boldsymbol{\alpha}} \rightleftharpoons \underline{\mathfrak{M}_{\triangleright}^{\circlearrowright\parallel}\lceil \boldsymbol{\alpha} \rceil}$ is connected with other informons via common operands and via operands appearing in common operand formulas, to a recursive depth, dimension, and circular perplexity.

An artificial entropon can be structured much more orderly, including a great number of well-structured formulas and formula systems, than a natural entropon which is mainly characterized by forgetting the well-formedness and particularities of formulas and formula systems. The artificial entropon can include a substantial number of already emerged informons, being "memorized" absolutely, that is, without loss of organization and meaning of informons, in the form they appeared concretely in certain situations. Within informational understanding, "memory" as a memorizing property means the circular structure of formulas and formula systems, representing a circular dynamic sort of memory, where the circular informing of entities (operands) is understood as an emerging process of memorizing in causal formula loops[5]. In contrast to informon, entropon is in general not absolutely or regularly structured in any way.

Similarly as for the terminology of informon in [20], Table 2, the additional terms for entropon can be introduced, to make the discussion concerning entropon transparent. By entropon, several words as the adjective *entroponic*, the adverb *entoponically*, the verb *entroponize*, and the participle *entroponizing* can be used.

An explicit verbal definition of entropon can be the following[6]:

[3]The quantum definitions of informon and entropon can be found on Internet in Cyrillic, for instance, in Елкин, С.В. и Д.А. Гаврилов. 1998. Информоны и энтропоны. At the web site ⟨http://www.aha.ru/∼latypov/informon.htm⟩. Other and the same Russian authors can be searched in English under Koulikov V. V., Yolkin S. V., Hariponov A. S., Gavrilov D. A., etc.

[4]Das Informon ist ein neuer Begriff für die Komplexität eines bewußten Seienden. Jedes Informon ist als eine informationelle Entität mit eigenem Bewußtsein beschafft. Ein individuelles Bewußtsein ist aus Informonen zusammengesetzt. Ein Informon ist im gegebenen Bewußtsein informationell verteilt und damit auch mehr oder weniger in anderen Informonen anwesend. Das Bewußtsein ist nichts anderes als ein informationelles Überlappen der Informonen und selbst ein Informon.

[5]This sort of dynamic memory conceptualism could come close to the quantum-mechanical phenomenology (informational dynamics) in brain, or at least support the concept theoretically.

[6]In German, the definition is: Das Entropon ist ein neuer Begriff für die Komplexität der Umgebung eines bewußten Seienden. Jedes Entropon ist als eine ungeordnete informationelle Entität aus gut-strukturierten und irregulären Teilen zusammengesetzt, entstehend durch das Informieren des Buwußtseinssystems. Ein individuelles Bewußtsein besteht aus Informonen und Entroponen als genannten Entitäten. Ein Entropon ist im Bewußtseinssystem informationell verteilt und damit mehr oder weniger in anderen Informonen und Entroponen anwesend. Ein System des Bewußtsens ist nichts anderes als ein informationelles Überlappen von Informonen und Entroponen und selbst ein Informon mit dem zugehörigen Entropon.

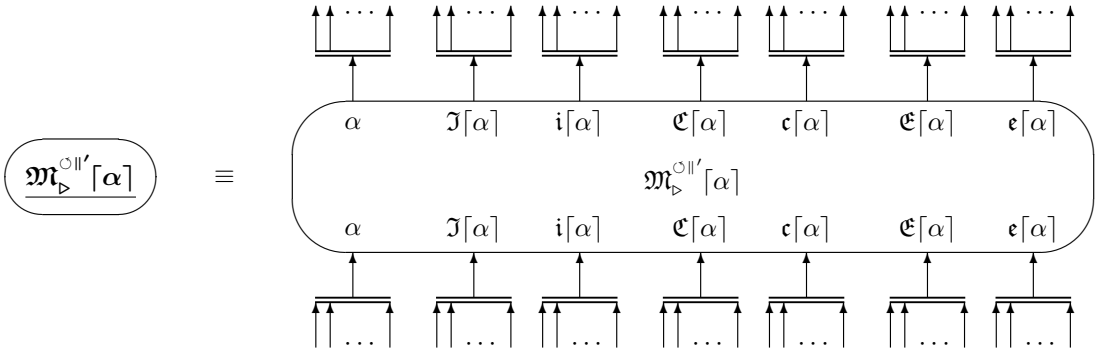


Figure 1: *The complexity of informon's primitive system* $\underline{\mathfrak{M}_{\triangleright}^{\circlearrowleft\!\shortparallel\prime}\lceil\boldsymbol{\alpha}\rceil}$ *is granted by parallel and serial connections to operands in the system around the primitive shell* $\mathfrak{M}_{\triangleright}^{\circlearrowleft\!\shortparallel\prime}\lceil\alpha\rceil$*, via the inner operands of name* α*, informing components* $\mathfrak{I}\lceil\alpha\rceil$ *and* $\mathfrak{i}\lceil\alpha\rceil$*, counterinforming components* $\mathfrak{C}\lceil\alpha\rceil$ *and* $\mathfrak{c}\lceil\alpha\rceil$*, and informational-embedding components* $\mathfrak{E}\lceil\alpha\rceil$ *and* $\mathfrak{e}\lceil\alpha\rceil$*.*

Entropon is a new concept for the complexity and disorder of a named conscious entity environment of an informon. Each entropon, as a disordered informational entity, constituted by well-formed as well as irregular informational items (in fact, parts of well-formed entities), is a consequence, arising by informing of a conscious system, consisting of informons and entropons. An individual consciousness consists of both informons and entropons as named entities. An entropon, in a given conscious system, is distributed informationally and, in this way, more or less, existent in other informons and entropons. A conscious system is nothing other than an informational overlapping of informons and entropons, and being an informon with the corresponding entropon by itself.

The question is, how the entropon can be defined formally, graphically, comparatively, and transparently to the concept of informon. The artificial entropon named α can be defined formally, for instance, as

$$\overline{\boldsymbol{\alpha}} \rightleftharpoons_{\mathsf{def}} \overline{\mathfrak{M}_{\triangleright}^{\circlearrowleft\!\shortparallel}\lceil\boldsymbol{\alpha}\rceil}, \text{ where } \triangleright \in \{\rightarrow, \leftarrow, \rightleftarrows, (\rightarrow, \leftarrow)\}$$

Here, $\mathfrak{M}_{\triangleright}^{\circlearrowleft\!\shortparallel}\lceil\alpha\rceil$ denotes the informonic shell in the form of metaphysicalistic decomposition of the name α, according to the subscript $\triangleright$. Because in the artificial informational environment,

$$\underline{\mathfrak{M}_{\triangleright}^{\circlearrowleft\!\shortparallel}\lceil\boldsymbol{\alpha}\rceil} \in \overline{\mathfrak{M}_{\triangleright}^{\circlearrowleft\!\shortparallel}\lceil\boldsymbol{\alpha}\rceil},$$

entropon $\overline{\mathfrak{M}_{\triangleright}^{\circlearrowleft\!\shortparallel}\lceil\boldsymbol{\alpha}\rceil}$, where $\triangleright \in \{\rightarrow, \leftarrow, \rightleftarrows, (\rightarrow, \leftarrow)\}$, can be divided in the sense of the set-theoretical view, in two areas, being separated by the metaphysicalistic shell $\mathfrak{M}_{\triangleright}^{\circlearrowleft\!\shortparallel}\lceil\alpha\rceil$, in fact, its longest loop. This loop can be grasped as the informonic envelope for which

$$\varphi_{\triangleright}^{\circlearrowleft}\lfloor\alpha, \mathfrak{I}\lceil\alpha\rceil, \mathfrak{i}\lceil\alpha\rceil, \mathfrak{C}\lceil\alpha\rceil, \mathfrak{c}\lceil\alpha\rceil, \mathfrak{E}\lceil\alpha\rceil, \mathfrak{e}\lceil\alpha\rceil\rfloor \in \mathfrak{M}_{\triangleright}^{\circlearrowleft\!\shortparallel}\lceil\alpha\rceil$$

The situation is presented in Fig. 2. The envelope is a formula of the metaphysicalistic shell—the initial inner conscious structure of the informon. Inside the envelope, the entire texture of the informon is meant to be located.

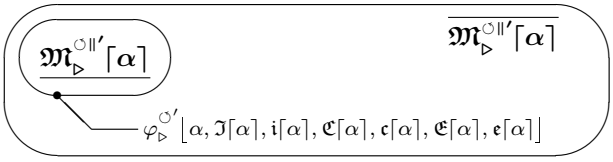


Figure 2: *In entropon's primitive system,* $\overline{\mathfrak{M}_{\triangleright}^{\circlearrowleft\!\shortparallel\prime}\lceil\boldsymbol{\alpha}\rceil}$*, informon's primitive system* $\underline{\mathfrak{M}_{\triangleright}^{\circlearrowleft\!\shortparallel\prime}\lceil\boldsymbol{\alpha}\rceil}$ *is informationally embedded. Envelope's primitive system* $\varphi_{\triangleright}^{\circlearrowleft\prime}\lfloor\alpha, \mathfrak{I}\lceil\alpha\rceil, \mathfrak{i}\lceil\alpha\rceil, \mathfrak{C}\lceil\alpha\rceil, \mathfrak{c}\lceil\alpha\rceil, \mathfrak{E}\lceil\alpha\rceil, \mathfrak{e}\lceil\alpha\rceil\rfloor$ *separates both and the contents of informon is meant to be encircled by the envelope.*

The shell delivers the basic metaphysicalistic organization to α (intentional, counter-intentional, and embedding-intentional informing), from which through complexity, that is, complex connectivism in the conscious system, informon as a conscious component can emerge. Informon's connectivism concerns all available operands in the system concerning informon's name directly and indirectly, associatively, and otherwise possibly. This situation is captured explicitly by Fig. 2.

One must not forget the correspondence between the formula φ, the formula scheme $\mathfrak{S}\lceil\varphi\rceil$, and the primitive system of the formula φ'. There is, $\varphi' \rightleftharpoons \mathfrak{G}\lceil\varphi\rceil$, where $\mathfrak{G}$ denotes the graph. For a formula system Φ, the correspondence becomes

$$\begin{array}{l}\Phi \longrightarrow \mathfrak{S}\lceil\Phi\rceil \longrightarrow \mathfrak{G}\lceil\Phi\rceil, \text{ with}\\ \Phi \rightleftharpoons (\varphi_1; \varphi_2; \ldots; \varphi_{n_\Phi}),\\ \mathfrak{S}\lceil\Phi\rceil \rightleftharpoons (\mathfrak{S}\lceil\varphi_1\rceil; \mathfrak{S}\lceil\varphi_2\rceil; \ldots; \mathfrak{S}\lceil\varphi_{n_\Phi}\rceil), \text{ and}\\ \Phi' \rightleftharpoons (\varphi'_1; \varphi'_2; \ldots; \varphi'_{n_\Phi})\end{array}$$

This correspondence is irreversible. From a primitive formula system Φ', the schematized system $\mathfrak{S}\lceil\Phi\rceil$ cannot be identified. And from a schematized system, the original formula system Φ cannot be identified. It seems that a natural conscious system operates schematically, however, an artificial conscious system can operate formula-likely,

schematically, and primitively, that is, graphically.

5 Cognitive-emotional nature of consciousness

The cognitive-emotinal concept of consciousness has to be understood merely as one of the important views in the philosophy of consciousness. Besides the cognitive-emotional paradigm, other explicit components distributed in a conscious system cooperate in the system orchestration, for instance, attention, sensibility, motivation, homeostasis, behavior, and the like. Cognition and emotion are fields researched extensively in the framework of informational phenomenalism, cognitive science, psychology, psychiatry, clinical research, biological systems, and so on.

5.1 Cognition and emotions

Cognition-driven emotional and emotion-driven cognitive systems exist within the natural and artificial consciousness. The aspect we discuss in this paper roots in the informational phenomenalism, where conscious components (like informons [19, 20]) interact in complex and interactive ways. Emotion-driven robots, like sociable and working robots, are examples of this sort of philosophy. Cognition-emotion perspective concerns different fields of research and clinical practice, that is, philosophy, psychology, psychiatry, cognitive science and the informational theory. The cognitive-emotional interconnection can be efficiently projected on the informational formalism, describing the most complex phenomena, and serving as an outlook for the informational consciousness implementation. The reader must not forget that the discussion in this paper is the subject of the aim dealing with the question, how to design a conscious system artificially in the future.

In nature, cognition and emotion are two of the key concepts in the constitution of conscious and subconscious (unconscious) domain, informing to and being informed by other subsystems. In the everyday language, reason and affect seem to support the main phenomenology of human consciousness. Scientifically, together with other significant components of the conscious system, cognitive-emotional perspective is dominating in complexity, interactivity, and specific (metaphysicalistic) organization of informing components. The following perspective, however not the complete one, is being worth to be considered, for instance, in robotics:

cognition	attention	homeostasis	behavior
emotions	motivation	sensorics	motorics

As the reader can guess, this list of conscious components can be used by the nowadays design possibilities of sociable robots, possessing only a not-yet complex constitution of each of the components.

5.2 Basic emotions

The question of basic emotions has its practical value in the design of emotion-driven and sociable robots. The criteria for basic emotions certainly can be set from different research points of view. Ekman ([8] pp. 55–56) lists 11 rules for distinguishing basic emotions from one another and from other affective phenomena. There are 15 distinguishable basic emotions:

> *amusement, anger, contempt, contentment, disgust, embarrassment, excitement, fear, guilt, pride in achievement, relief, sadness/distress, satisfaction, sensory pleasure,* and *shame.*

Each of these words denotes a family of related emotions, where some affective phenomena are omitted. Guilt may be not a basic emotion when considering its affective components. Interest may not be an emotion but rather a cognitive state.

According to Ekman's classification of basic emotions, romantic love, parental love, and hate are clearly affective, as is grief, and jealousy. They should belong to the so-called *emotional plots,* in which a number of basic emotions can be expected. Emotional plots inform in a more enduring and specific ways than basic emotions, and they can occur in specific affective contexts. The moods are another affective phenomena having different causes, lasting much longer, and being saturated with emotions. Affective personality traits, such as hostility, are another set of affective phenomena.[7]

It is hard to say how many emotions there are, however, linguistically many words naming emotions in different languages and cultures exist. The meaning of these words can be understood as features of eliciting situations and of different responses to reality and phenomenal happening. The name of an emotion, its linguistic denotation, and its meaning is language-conditioned and depends on the development and spreading of the language. Highly developed languages distinguish a high number of different emotions, e.g. in English some 2,500.

[7] A transparent historical phenomenon of emotional plots is the so-called class hate and class hostility, elapsing in crime upon humanity and class genocide, in totalitarian systems of communism. On the other side of such a hate, the love to the totalitarian political party is demonstrated, and experienced in the highest possible affective state of excitement, pride, and satisfaction. For instance, a communist youngster confessed that she was extremely excited, with tears in her eyes, when being accepted as a member of communist party. Today, as an old philosopher, she is still active in the leftist action and organization, provoking the normally emerging democratic public. This situation may prove how her emotional plot is enduring and lasting over the real possibilities of life. In this sense, a totalitarian ideology is one of the best examples of affective revolutionary conspiracy—a totalitarian plot.

In this respect it can be useful to remind the usual meaning of the word *plot.* Plot is a secret plan for accomplishing an evil or unlawful end, synonymous with intrigue, machination, cabal, conspiracy, scheme. It is a design, plan, trick, contraption, maneuver, stratagem and the like.

5.3 Basic components of cognitive-emotional complex

Because of their informational perplexedness, it is not possible to separate and distinguish strictly cognitive and emotional components. Cognition is triggered by a kind of affective component and, then, being cognitively coupled with it. A first-step approximative approach to the design of cognitive-emotional system can be realized via the following list of characteristic signifiers (names):

ability	affect	affection	anger
anticipation	antipathy	anxiety	apathy
appeal	attention	aversion	avoiding
awareness	believing	bravery	conviction
convincing	courage	delight	desire
despair	disability	disliking	distress
disturbance	emotion	emotionalizing	empathy
empathizing	enjoyment	ethics	evading
excitement	fear	feeling	frightening
grief	hate	hope	horror
impression	inclination	instinctiveness	intention
intuitiveness	joy	liking	love
lust	motivation	opinion	pain
passion	perceiving	pity	pleasure
($\acute{\eta}\delta o\nu\acute{\eta}$)	presentiment	propaganda	provocation
rage	recognition	repugnance	sadness
seeming	sense	sensation	sensing
sensitivity	sentiment	sorrow	($\lambda\upsilon\pi\eta$)
subjectivism	susceptibility	suspicion	sympathy
tending to	touching	trusting	truthfulness
turgidness	turmoil	twitting	understanding

This list of affect and cognition components, that is, of cognitive-emotional signifiers, can serve as possibilities to the initial impulse of design using the principles of consciousness (informonic) decomposition. Each of the components can be differently decomposed according to the available knowledge, information, and data archives including signifier definitions, being connected with other components via common operands.

Corresponding lists of signifiers can be completed for other domains of consciousness components, for the fields of reason, understanding, language, and other domains of consciousness informational activity. For the decomposition of reason, different signifiers can be useful, for instance:

argument	awareness	cause	comprehension
concluding	grasping	ground	intellect
intelligence	legitimacy	judgment	justice
logic	mind	motif	perception
proof	reason	reasoning	right
theme	thinking	tolerance	understanding

Further, for the decomposition of understanding, still additional signifiers come into the foreground:

being conscious	conceiving	conceptualizing	interpreting
perceiving	reasoning	sensing	understanding

Sensuality, attention, motivation, cognition, emotions, homeostasis, behavior, and additional components can be added to a conscious system, performing in the informational and physical reality. In designing the artificial conscious system, an exhaustive list of components, constituting the cognitive-emotional complex, can be helpful, as presented in Table 1. All this could represent a rough approximation of the nature of components entering into the context of conscious systems.

Homeostasis or *homeostatic informational equilibrium* of a conscious entity informing is the relative stability (perseverance of intention), for instance, of the internal organization of informon, by circular (feedback, loop) mechanisms, despite the presence of inner and outer impulses (e.g., ideological, dogmatic, prejudiced disturbances), capable of causing profound changes. Let us analyze and show some possibilities of the attention organization in the next section.

6 The problem of conscious and unconscious attention

Attention belongs to the basic concerning a conscious system. Thus, attention is a component, which together with cognition, emotions, motivation, homeostasis, arousal, behavior, etc., constitutes the conscious system, as imagined today in the research of natural consciousness [6, 16], and being actual at the design of artificial conscious devices, like robots and informational systems dealing with phenomenal organization of systems. Informational formulas, informational schemes, and informational graphs are important tools of design in natural and artificial systems possessing the conscious function. Especially, graphs offer a general overview of a situation concerning a conscious system, putting into the foreground the circular system or subsystem organization, and the informational perplexedness in the graph. In this sense, a complex graph can represent the most advanced field for informational experiments [21]. From a graph, new schemes can be constructed when moving along the graph paths. Thus, the irreversibility of a graph to the original schemes, which originated its perplexed structure, is the condition for a new design within a complex conscious system.

Let us present in more detail the interinforming of *attention,* $\mathfrak{a}_{\text{attention}}$, with other entities (cognition, emotions, etc.), as an informonic-entroponic organized system $\left(\underline{\mathfrak{a}}_{\text{attention}}; \overline{\mathfrak{a}}_{\text{attention}}\right)$. One of the main problems is, how to organize the complex function of *attention* in a conscious system. There is more or less evident that the attention to a certain contents (object) will be triggered by intrinsic and outer impulses, originating in informational happening. The inner sensuality and the events happening in the system environment will deliver the triggering information to the *system of attention.*

Attention, $\mathfrak{a}_{\text{attention}}$, regulates the extent, to which a conscious system $\left(\underline{\mathfrak{z}}; \overline{\mathfrak{z}}\right)$ is distracted from aversive stimuli, with the aim, to replace the unwanted emotions. Entropon $\overline{\mathfrak{z}}$ has the role of subconscious or unconscious sys-

Table 1: *A short ad-hoc catalog of perplexed cognition and emotion concerning components (more in [21], Subsubsect. 27.4.1, Tab. 17).*

tem within $(\underline{\mathfrak{z}}; \overline{\mathfrak{z}})$. The part of attention is, in a circular way, a regulatory informing of emotional behavior, $\mathfrak{I}\lceil \mathfrak{r}_{\text{regulation}} \lceil \mathfrak{b}_{\text{behavior}} \lceil \mathfrak{e}_{\text{emotions}} \rceil \rceil \rceil$. In human, the affect-relevant *attentional informing*, $\mathfrak{I}\lceil \mathfrak{a}_{\text{attention}} \lceil \mathfrak{a}_{\text{affect\_relevant}} \rceil \rceil$, is under control of prefrontal cortex [7]. This situation can be expressed transparently by the formal language of informational schemes, in the form of the scheme,

$$\mathfrak{p}_{\text{prefrontal\_cortex}} \models_{\text{control}} \mathfrak{I}\lceil \mathfrak{a}_{\text{attention}} \lceil \mathfrak{a}_{\text{affect\_relevant}} \rceil \rceil$$

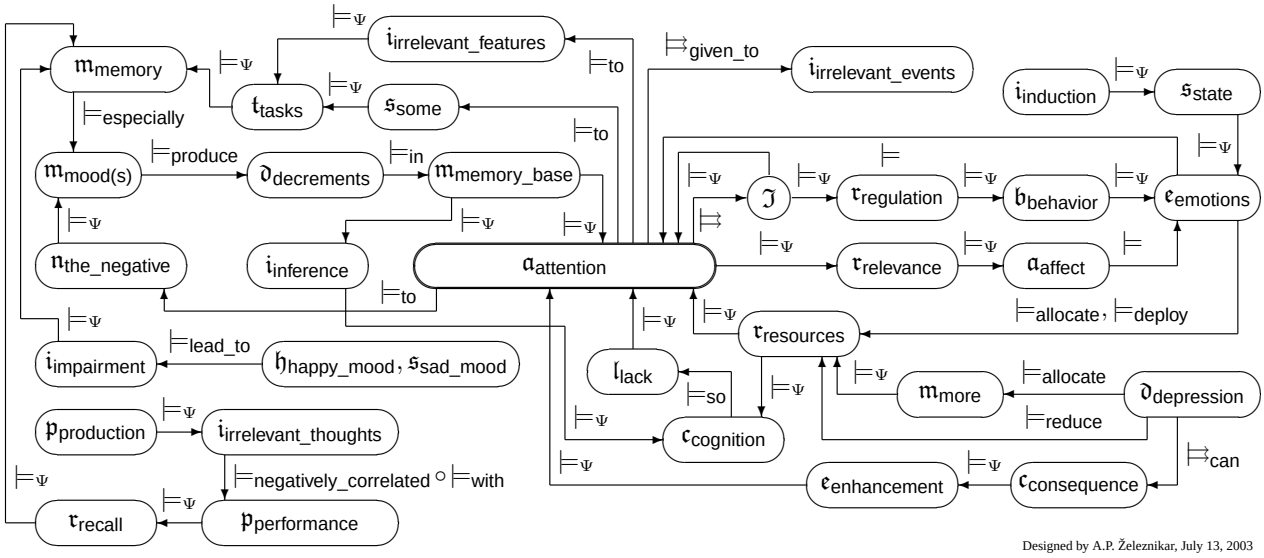


Figure 3: *The informationally modified Ellis and Moore subsystem of attention [9], concerning mood, memory, depression, irrelevant events, thoughts, and features. Also,* $\mathfrak{a}_{\text{attention}} \rightrightarrows \mathfrak{I}\lceil \mathfrak{r}_{\text{regulation}}\lceil \mathfrak{b}_{\text{behavior}}\lceil \mathfrak{e}_{\text{emotions}}\rceil\rceil\rceil \models \mathfrak{a}_{\text{attention}}$.

The expression $\mathfrak{I}\lceil \mathfrak{a}_{\text{attention}}\lceil \mathfrak{a}_{\text{affect\_relevant}}\rceil\rceil$ on the right side of operator $\models_{\text{control}}$ can be read as (1) the affect-relevant attentional informing, (2) informing of attention being affect-relevant, (3) informing concerning attention of the affect relevant domain, etc.

Mood produces decrements in memory base concerning attention and cognitive inference, schematically,

$$\mathfrak{m}_{\text{mood}} \models_{\text{produce}} \mathfrak{d}_{\text{decrements}} \models_{\text{in}} \mathfrak{m}_{\text{memory\_base}}\lceil \mathfrak{a}_{\text{attention}}, \mathfrak{i}_{\text{inference}}\lceil \mathfrak{c}_{\text{cognition}}\rceil\rceil$$

Induction of an emotional state will allocate or deploy *attentional resources* to some memory task,

$$\left(\mathfrak{i}_{\text{induction}}\lceil \mathfrak{s}_{\text{state}}\lceil \mathfrak{e}_{\text{emotion}}\rceil\rceil \left(\begin{matrix}\models_{\text{allocate}}\\ \models_{\text{deploy}}\end{matrix}\right) \mathfrak{r}_{\text{resources}}\lceil \mathfrak{a}_{\text{attention}}\rceil\right) \models_{\text{to}} \mathfrak{s}_{\text{some}}\lceil \mathfrak{t}_{\text{tasks}}\lceil \mathfrak{m}_{\text{memory}}\rceil\rceil$$

Depression allocates more attentional resources to irrelevant features of the memory task, especially moods, and reduces the cognitive resources, so the lack of attention is given to relevant events. Schematically,

$$\left(\begin{array}{l}\mathfrak{d}_{\text{depression}} \models_{\text{allocate}} \mathfrak{m}_{\text{more}}\lceil \mathfrak{r}_{\text{resources}}\lceil \mathfrak{a}_{\text{attention}}\rceil\rceil \models_{\text{to}} \\ \quad \mathfrak{i}_{\text{irrelevant\_features}}\lceil \mathfrak{t}_{\text{tasks}}\lceil \mathfrak{m}_{\text{memory}}\rceil\rceil \models_{\text{especially}} \mathfrak{m}_{\text{moods}}; \\ \mathfrak{d}_{\text{depression}} \models_{\text{reduce}} \mathfrak{r}_{\text{resources}}\lceil \mathfrak{c}_{\text{cognition}}\rceil \models_{\text{so}} \\ \quad \mathfrak{l}_{\text{lack}}\lceil \mathfrak{a}_{\text{attention}}\rceil \rightrightarrows_{\text{given\_to}} \mathfrak{r}_{\text{relevant\_events}}\end{array}\right)$$

Both happy and sad moods lead to *memory impairment*, that is,

$$\mathfrak{h}_{\text{happy\_mood}}, \mathfrak{s}_{\text{sad\_mood}} \models_{\text{lead\_to}} \mathfrak{i}_{\text{impairment}}\lceil \mathfrak{m}_{\text{memory}}\rceil$$

Depression can be a consequence of enhanced attention to negative moods, that is, schematically,

$$\mathfrak{d}_{\text{depression}} \rightrightarrows_{\text{can}} \mathfrak{c}_{\text{consequence}}\lceil \mathfrak{e}_{\text{enhancement}}\lceil \mathfrak{a}_{\text{attention}}\rceil\rceil \models_{\text{to}} \mathfrak{n}_{\text{the\_negative}}\lceil \mathfrak{m}_{\text{moods}}\rceil$$

Production of irrelevant thoughts is negatively correlated with recall performance, that is,

$$\mathfrak{p}_{\text{production}}\lceil \mathfrak{i}_{\text{irrelevant\_thoughts}}\rceil \rightrightarrows_{\text{negatively\_correlated}} \circ \models_{\text{with}} \mathfrak{p}_{\text{performance}}\lceil \mathfrak{r}_{\text{recall}}\lceil \mathfrak{m}_{\text{memory}}\rceil\rceil$$

This completes the concepts given by Ellis and Moore [9], presented by the informational graph in Fig. 3.

The graph in Fig. 3 needs a commentary. The path $\alpha \models_{\Psi} \beta$ is read α *concerns* β or, colloquially, α *of* β, or, in the adjectival style, α*-like* β or β*-like* α, e.g., cognitive inference instead of inference concerning cognition or, in adjectively used noun β, $\beta\alpha$, e.g., memory base for base of memory. Instead of irrelevant thoughts, thoughts of the irrelevant (domain) could be chosen. In the text discussed, formula schemes are lost in the graph being a union of different schemes and bringing to the surface a scheme-transitive circular organization. Isolated circularly, terminal entities in Fig. 3 are *production of irrelevant thoughts,* $\mathfrak{p}_{\text{production}}\lceil \mathfrak{i}_{\text{irrelevant\_thoughts}}\rceil$, *depression,* $\mathfrak{d}_{\text{depression}}$, and the *happy* and *sad mood,* $\mathfrak{h}_{\text{happy\_mood}}$ and $\mathfrak{s}_{\text{sad\_mood}}$, respectively.

A distinction in *attention informing* between the automatic and conscious, or automatic and controlled, can be introduced. *Attention* as an informational entity is organized in a flexible way and distributed across stimuli and tasks. *Automatic attention* is concurrent and without mutual interference (an informon $\underline{\mathfrak{a}_{\text{automatic\_attention}}}$). *Conscious attention* is controlled by time-sharing of tasks. *Awareness* of automatic informing is low and high for controlled informing. *Attentional system* is related to unconscious and conscious informing. The modern concept of *working memory* is related to conscious informing. *Noticing* refers to the introspective awareness of perceiving (acknowledging, recognizing, observing). *Selective attention* is a component in the transformation of a perception into

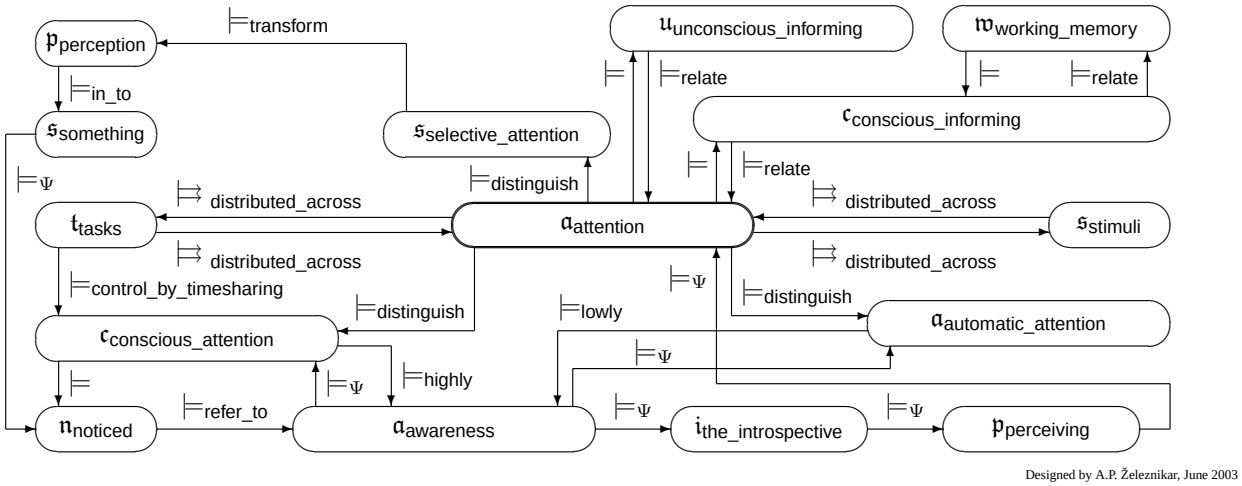


Figure 4: *Informationally modified Öhman's subsystem of attention [17] within a perceptive-emotional organization of a conscious system.*

something noticed. All this, extracted through Öhman's diction [17], can be summarized by the system graph in Fig. 4.

We see how a short text (paragraph) in the natural language generates the graph in Fig. 4 being informationally complemented, delivering a relatively complex and circularly structured subsystem graph concerning $\mathfrak{a}_{\text{attention}}$. Together with the initial metaphysicalistic subsystem given by $\mathfrak{M}_{\triangleright}^{\circlearrowleft\!\!\parallel}\lceil\mathfrak{a}_{\text{attention}}\rceil$ and the informonic extension into the informational space, informon $\underline{\mathfrak{a}_{\text{attention}}}$ can emerge and come to a temporary existence. It emerges within and out of its entropon $\overline{\mathfrak{a}_{\text{attention}}}$.

On the way to a complete frame definition of attention, several items from the cognitive-emotional domain of research can be added [6, 16] and formalized by informational schemes and graphs. Problems of *attention* and *memory* are highly distractible and play a specific role in manic and paranoid communication. Cognitive informing involved in mania, like that involved in paranoia, has the same informational ground with that implicated in depression. Mania is an extreme state of depression and performs as an extreme defense against depression. Manic episodes (communication) progress through euphoria, anger, irritability and, lastly, through depression, panic and delirium [3].

Attention effects appear to be more easily obtained in anxiety disorders, while depressive states seem to influence memory rather than attention [10]. Forgetting of essential history, trauma and experience seems to be a depressive phenomenon.

Emotional arousal directs *attentional* informing to features which are central to the arousal, at the expense of peripheral details, e.g., in case of eyewitness. Differential attentional informing can occur in response to the negative features of the event [1].

Attention to the environment is the very beginning property of conscious system. Through stimuli, basic emotions emerge and are learned from the environment. The organization of attention, informing on the path from perceptual systems to the contents of high-level reasoning, is interrelated with emotional and cognitive states of the system, where positive emotions can broaden attentional focus. In fear, shifts in perception and attention appear, goals and motivational weightings change, attentional informing is redirected, conceptual frames change in the direction "dangerous" or "safe", memory informing steps to new retrieval tasks, communication processes change, special inference and learning systems are activated, physiology changes, and behavioral decision rules are activated [5].

Attention can be *selectively dependent on interest*, so it does not stray randomly through the vast domain of stimuli impinging on the senses. Interest focuses attention on a particular object, and as an emotion provides the motivation, interaction, constructive and creative endeavor, and the development of intelligence [13].

It seems that sadness impairs *attention* to tasks. Sadness is associated with *lowering of attention*, where sadness is not being caused by another agent, and so attention is focused inward [2].

The discussed organization and interweavement of attention with other components can now be considered with the aim to make the graph in Fig. 4 more complex and, at last, construct the attention informon, and identifying the existent attention entropon, including the meaningly, associatively and otherwise related words, word phrases, nouns, verbs, formulas, schemes, graphs, etc., which will impact the instantaneous emergence of the attention informon.

7 Implementation of informational phenomenalism

Informational phenomenalism is the metaphor for the conscious system, emerging dynamically, in the live and the artificial. Within this texture, informonic and entroponic entities function as conscious and subconscious (unconscious)

components per se, respectively, in the complex informational space concerning the individual and global conscious system. The condition sine qua non of a component *to be conscious* is the complexity of already existing conscious environment, found in a conscious system, and component's initial metaphysicalistic organization, enabling the emergent development of conscious and self-conscious (metaphysicalistically phenomenal) abilities. Such components' organization and components' environment guarantee the conscious performance of informational entities. Informons together with entropons constitute the conscious system on the conscious and subconscious level, respectively, in local and global circumstances.

Implementing informational phenomenalism means to begin with the initial organization of a couple of informons, realizing the informonic *interism* in the sense pointed out in Sect. 1. In the initial state, a component can gain the property of being conscious through other already conscious components, by which it is linked through common operands. In this manner, the mechanism of the so-called conscious bootstrapping can perform at once, after a short developmental period, or by endurance, developing isolated from the environment in the original initial way of conceptualization. This sort of development is the consequence of system's decompositions concerning the initial entities, including the metaphysicalistic and other sorts of informational decomposition. Decompositions by themselves are conscious entities with their own informonic and entroponic structure and organization within the conscious system.

At last, let us point to the possibilities, how to use an informational graph for construction of sentences in English, when moving along the linked graph paths. Let us take the scheme from the graph in Fig. 4,

$$\mathfrak{s}_{\text{selective\_attention}} \models_{\text{transform}} \mathfrak{p}_{\text{perception}} \models_{\text{in\_to}} \mathfrak{s}_{\text{something}} \models_{\Psi} \mathfrak{n}_{\text{noticed}} \models_{\text{refer\_to}} \mathfrak{a}_{\text{awareness}} \models_{\Psi} \mathfrak{i}_{\text{the\_introspective}} \models_{\Psi} \mathfrak{p}_{\text{perceiving}} \models_{\Psi} \mathfrak{a}_{\text{attention}}$$

The possible parenthesizing of the scheme is, for instance, the well-formed formula,

$$\Big(\big((\mathfrak{s}_{\text{selective\_attention}} \models_{\text{transform}} \mathfrak{p}_{\text{perception}}) \models_{\text{in\_to}} (\mathfrak{s}_{\text{something}} \models_{\Psi} \mathfrak{n}_{\text{noticed}})\big) \models_{\text{refer\_to}} \big((\mathfrak{a}_{\text{awareness}} \models_{\Psi} \mathfrak{i}_{\text{the\_introspective}}) \models_{\Psi} \mathfrak{p}_{\text{perceiving}}\big)\Big) \models_{\Psi} \mathfrak{a}_{\text{attention}}$$

One of the possible readings of this formula could be the following: *Concerning attention, selective attention transforms a perception, into something noticed, referring to the introspective awareness of perceiving.* An exact miming of the formula parenthesizing in a natural language is in many cases impossible. Thus, a sentence remains a compromise between an informational scheme and an informational formula (see informational experiments in [21]).

8 Conclusion

The informon-entropon debate can now be made more transparent through the case of attention, discussed in Sect. 6. In the beginning, the name *attention* as operand $\mathfrak{a}_{\text{attention}}$ comes fore, for which the meaning is searched, for instance, in the professional literature. In fact, the fragments of possible meaning for the concept named *attention* are gathered within an entity. In the formalized form, this emergent entity of found concepts can be understood as entropon $\overline{\mathfrak{a}_{\text{attention}}}$. One of the concepts being included in $\overline{\mathfrak{a}_{\text{attention}}}$ has the graph, according to Ellis and Moore [9], presented in Fig. 3. The searching for possible concepts continues with loading of formalized results to entropon $\overline{\mathfrak{a}_{\text{attention}}}$. One of such cases is presented, according to Öhman [17], by the graph in Fig. 4. From the text of Sect. 6, which was not exhausted entirely, further formalized concepts concerning attention can be mined. Further concepts for attention, as a function within the conscious system, can be found elsewhere (in [6, 16], and the references herein, for instance). In this way, entropon $\overline{\mathfrak{a}_{\text{attention}}}$ can be gradually constituted to an enormous extent of complexity. Many new operands appear and serve simultaneously as common operands within different concepts of attention.

The constitutive step in emerging of informon $\underline{\mathfrak{a}_{\text{attention}}}$ can now begin through the initial metaphysicalistic decomposition of the name $\mathfrak{a}_{\text{attention}}$, that is, $\mathfrak{M}_{\triangleright}^{\circlearrowleft\|}\lceil\mathfrak{a}_{\text{attention}}\rceil$. The common operand to the entities of entropon $\overline{\mathfrak{a}_{\text{attention}}}$ is certainly $\mathfrak{a}_{\text{attention}}$. In general, by its subsystems, decomposition $\mathfrak{M}_{\triangleright}^{\circlearrowleft\|}\lceil\mathfrak{a}_{\text{attention}}\rceil$ means attentional informing, attentional counterinforming, and attentional informational embedding, as known from the general metaphysicalistic organization of entities [18]–[21]. Then, decomposition $\mathfrak{M}_{\triangleright}^{\circlearrowleft\|}\lceil\mathfrak{a}_{\text{attention}}\rceil$ can be connected with items residing in entropon $\overline{\mathfrak{a}_{\text{attention}}}$, as shown in Fig. 1, in a meaningly reasonable way. It has to be analyzed, which entities fit the meaning of attentional informing and where the connection from and to $\mathfrak{I}\lceil\mathfrak{a}_{\text{attention}}\rceil$ and $\mathfrak{i}\lceil\mathfrak{a}_{\text{attention}}\rceil$ of $\mathfrak{M}_{\triangleright}^{\circlearrowleft\|}\lceil\mathfrak{a}_{\text{attention}}\rceil$ can lead. Meaningly reasonable connection can be realized for metaphysicalistic operands of attentional counterinforming, $\mathfrak{C}\lceil\mathfrak{a}_{\text{attention}}\rceil$ and $\mathfrak{c}\lceil\mathfrak{a}_{\text{attention}}\rceil$, and attentional informational embedding, $\mathfrak{E}\lceil\mathfrak{a}_{\text{attention}}\rceil$ and $\mathfrak{e}\lceil\mathfrak{a}_{\text{attention}}\rceil$, leading to and from the entities of the entroponic environment.

In the sketched way, the informon $\underline{\mathfrak{a}_{\text{attention}}}$ emerges through the initial step of complexity given by decomposition $\mathfrak{M}_{\triangleright}^{\circlearrowleft\|}\lceil\mathfrak{a}_{\text{attention}}\rceil$, and through that, to what it can be connected in entropon $\overline{\mathfrak{a}_{\text{attention}}}$. Through more and more connections and informational perplexity, considering the interinformonic and interentroponic connections, the complexity becomes enormous and, by this, the informon $\underline{\mathfrak{a}_{\text{attention}}}$ performs, from the step of sufficient complexity on, as a conscious (and self-conscious) informational entity, within a certain conscious subject, the natural or the artificial one. Metaphysicalism and complex connectivity lead to the property of conscious informing of entity within a conscious informational system. In this manner, emotions, cognition, motivations, goals, decisions, and so

on, emerge to conscious entities.

Throughout this paper, the reader is confronted with the natural and formal-informational language — the transformation from the natural language to the so-called $\mathfrak{Z}$-language[8] [21] and vice versa. Using the natural language, we know of what we speak about, become aware of the object, through the informing of word sequences, describing and interpreting it. In this view, the consciousness concerning something is language dependent, inclusive the cognitive-emotional perspective by itself; it is a constitutive part of the language and nothing more. Thus, consciousness, cognitions, emotions, motivations, attentions, behaviors, etc. depend, emerge, and come to consciousness strongly and substantially through language, its informational development, emergence, fitting, and by producing meaning to the occurring conscious situation.

The name α of informon $\underline{\boldsymbol{\alpha}}$ and entropon $\overline{\boldsymbol{\alpha}}$, in the named informational space $\overline{\underline{\boldsymbol{\alpha}}}$, is functioning like the intention, supervising and forcing the production (informational emerging) of meaning, closely and distantly concerning α. Through α-intentional informing, α-counterinforming, and α-informational embedding, a possible, sufficiently exhaustive, reasonable, and complex interpretation of α can emerge, ending in the interaction with other informons and entropons of the conscious system, in the conscious and subconscious (unconscious) property of the α-informon and α-entropon, respectively. At different informational situation, new informons and entropons can emerge, carrying a different meaning, fitting best the new informational situation. This sort of chaotic informing, concerning a definite name α, happens in everyday informing in a live, individual conscious system. In an artificial conscious system, the emerging of α's meaning may be more precise, disciplined and stable, and certain in this way. Artificial consciousness, through its informons and entropons, can emerge to a certain and disciplined mind, being dedicated to specific problems and themes of informational actuality.

References

[1] BEKERIAN, D.A. & S.J. GOODRICH. 2000. Forensic applications of theories of cognition and emotion. *In* [6] pp. 783–798.

[2] BARR-ZISOWITZ, C. 2000. "Sadness" — Is there such a thing? *In* [16] 607–622.

[3] BENTALL, R.P. & P. KINDERMAN. 2000. Self-regulation, affect and psychosis: The role of social cognition in paranoia and mania. *In* [6] pp. 353–381.

[4] BUTTAZZO, G.C. 2001. *Artificial consciousness: Utopia or real possibility?* IEEE Computer 34:7:24–30.

[5] COSMIDESS, L. & J. TOOBY. 2000. Evolutionary psychology and the emotions. *In* [16] 91–115.

[6] DALGLEISH, T. & M. POWER, EDS. 2000. Handbook of Cognition and Emotion. John Wiley & Sons. Chichester, England.

[7] DAVIDSON, R.J. 2000. Neuropsychological perspectives on affective styles and their cognitive consequences. *In* [6] 103–123.

[8] EKMAN, P. 2000. Basic emotions. *In* [6] 45–60.

[9] ELLIS, H.C. & B.A. MOORE. 2000. Mood and memory. *In* [6] 193–210.

[10] FORGAS, J.P. 2000. Network theories and beyond. *In* [6] 591–611.

[11] HEIDEGGER, M. 1962. Being and Time. Translated by J. Macquarrie & E. Robinson. Harper & Row. New York.

[12] HEIDEGGER, M. 1986. Sein und Zeit. Sechzehnte Auflage. Max Niemeyer Verlag. Tübingen.

[13] IZARD, C.E. & B.P. ACKERMAN 2000. Motivational, organizational, and regulatory functions of discrete emotions. *In* [16] 253–264.

[14] KURZWEIL, R. 1999. The Age of Spiritual Machines. When Computers Exceed Human Intelligence. Penguin Books. New York.

[15] LAZARUS, R.S. 2000. The cognition-emotion debate: A bit of history. *In* [6] 3–19.

[16] LEWIS, M. & J.M. HAVILAND-JONES, Eds. 2000. Handbook of Emotions. Second Edition. The Guilford Press. New York, London.

[17] ÖHMAN, A. 2000. Distinguishing unconscious from conscious emotional processes: Methodological considerations and theoretical implications. *In* [6] 321–352.

[18] ŽELEZNIKAR, A.P.[9] 1998. Topological informational spaces. Informatica 22:287–308.

[19] ŽELEZNIKAR, A.P.[9] 2002. *Informon — ein bewusster Bauteil des Bewußtseins.* Grundlagenstudien aus Kybernetik und Geisteswissenschaft — Humankybernetik 43:4:153–161.

[20] ŽELEZNIKAR, A.P.[9] 2002. *Informon — An emergent conscious component.* Informatica 26:419–431.

[21] ŽELEZNIKAR, A.P.[9] 2003. Introduction to Artificial Consciousness. The Philosophy of the Informational, Formalization, and Implementation. A study in progress.

[8] Symbol $\mathfrak{Z}$ stands for the consciousness informing within the metaphysicalistic decomposition $\mathfrak{M}_{\triangleright}^{\langle\rangle\|}\lceil\mathfrak{z}\rceil$, where $\mathfrak{z}$ is the designator for $\mathfrak{c}$consciousness·

[9] In PDF, at <www.artifico.org>.

A Comparison Between Exact and Approximate Method for Solution of General One-Dimensional Cutting Stock Problem

Peter Trkman and Miro Gradišar
Faculty of Economics
Kardeljeva ploscad 17
1000 Ljubljana
Slovenia

Keywords: cutting, optimisation, branch & bound, decision tree

Received: July 17, 2003

The paper describes exact solution of general one-dimensional cutting stock problem (G1D-CSP) where all stock lengths are different. Branch & Bound (B&B) optimization method is used. The solution is cutting plan with minimized overall trim loss in such a way that order lengths are cut in exactly required number of pieces and only one stock length is in general not cut to the end. If it's long enough then it can be used later and is not treated as a waste. G1D-CSP can also be solved approximately with Sequential Heuristic Procedure (SHP). Comparison between B&B and SHP is presented. It is shown that exact solution is better when the size of the problem does not exceed certain limit. The question is, how to determine this limit, which can be different in different practical situations. An approach, based on decision trees, for the selection of appropriate method for each individual case, is proposed. Numerous examples are calculated.

1 Introduction

One-dimensional stock cutting occurs in many industrial processes [4,8,9,16,19] and during the past few years it has attracted increased attention of researchers from all over the world [1,2,11,12,18,21]. Most standard problems related to one-dimensional stock cutting are known to be NP-complete and in general a solution can be found by using approximate methods and heuristics. However, the unbelievable development of computers and constantly growing processing power are pushing the complexity limit of the cutting problems, where exact methods could be used, slightly up. Therefore importance of exact methods is growing and the number of practical situations, where they can be used, increases [8,14, 18].

Dyckoff [3] classifies the solution of cutting stock problems into two groups: *pattern oriented* and *item-oriented*. In the *pattern-oriented* approach, at first, order lengths are combined into cutting patterns, for which - in a succeeding step - the frequencies, that are necessary to satisfy the demands, are determined. This approach can be applied when the stock is of the same length [6] or of few groups of standard lengths [7]. *Pattern-oriented* approach is most common in practice and therefore the type of the problem we get in this case is designated as Standard One-Dimensional Cutting Stock Problem (S1D-CSP) [5,12,13]. To solve (S1D-CSP) the classic LP-based Gilmore and Gomory's "delayed pattern generation" [6,7] or any other LP-based Method (LPM) is mostly used [16,18,19,22,23].

The *item-oriented* approach is characterized by individual treatment of every item to be cut. If stock lengths are all different then frequencies others than 0 and 1 cannot be determined and only an *item-oriented* solution can be found [9,10].

An *item-oriented* approach is more general and can be theoretically applied regardless of the *assortment of large objects* [3,12,13]. Therefore the problem we get in this case is designated as General One-Dimensional Cutting Stock Problem (G1D-CSP) [11,13]. There are two possibilities to solve G1D-CSP: exact methods (branch and bound, dynamic programming) or approximation algorithms in form of SHP [20]. SHP seems to be better regarding robustness and potential usefulness in a wide range of cases. Also time complexity of SHP is in general lower. On the other hand exact solution is better where the size of the problem doesn't exceeds acceptable limits and becomes intractable. But even in such cases approximate solution obtained in some acceptable time period, as a temporary result of exact method, can be comparable with the one of SHP.

Many examples of exact methods for solving different types of cutting problems are described in literature [8,18]. But we didn't found any exact solution of G1D-CSP so far.

If there are many methods available for the solution of G1D-CSP, then it can be difficult to select the right one. Therefore we decided to propose an approach based on decision trees for the selection of suitable solution method, which will take in the account the problem size, the quality of the solver and the computer speed.

This paper is organized as follows: in next section the definition of cutting problem is given. Exact solution development, using Branch & Bound optimization method in the form of a computer program,

is presented with numerous practical examples. In section 4 those results are compared with the results, acquired with SHP. Finally a new approach, based on decision trees for the selection of suitable solution method for each individual case, is proposed. It is shown that this approach can indeed lead to reduced trim loss.

2 Problem definition and solution method

G1D-CSP is one-criterial minimization problem. The criterion is overall trim loss. G1D-CSP satisfies two general conditions:

1. If there is abundance of material order lengths are cut in exactly required number of pieces (In S1D-CSP this number can be greater than demanded.).
2. Only one stock length cannot be cut to the end. The result is residual length that can be used later (In S1D-CSP all used stock lengths are cut to the end.).

Details in problem definition are similar to those in [12] and [10]. The difference regarding [12] is that we also included orders that cannot be fulfilled entirely due to shortage of material in stock. The difference regarding [10] is that distribution of uncut pieces by order lengths is not included. It is more difficult to control factors other than trim loss with exact methods than with SHP. Other differences regarding both [12] and [10] doesn't affect the content of the problem but only the way of expression, which is adapted to the solution method.

First we need to decide whether we will use branch and bound method or some dynamic programming. Branch and bound (B&B) exact method was chosen. There are two reasons for that. First B&B is standard method and second, the market is offering many OR computer packages with B&B. Some of them allow using B&B as a subroutine. This means that it could be included in some application program. In our case the application program collects data, checks whether there is an abundance or shortage of material, solves the appropriate model and displays the results.

Problem is defined as follow:

For every customer order a certain number of stock lengths is available. In general all stock lengths are different. We consider the lengths as integers. If they are not originally integers we assume that it is always possible to multiply them with a factor and transform them to integers. It is necessary to cut a certain number of order lengths into required number of pieces. The following notation is used:

s_i = order lengths; $i = 1,...,n$.
b_i = required number of pieces of order length s_i.
d_j = stock lengths; $j = 1,...,m$.
x_{ij} = number of pieces of order length s_i having been cut from stock length j.
y_j indicates whether the stock length j is used in the cutting plan (y_j=1 if stock length j is not used in the cutting plan).
u_i indicates whether the remainder of stock length j is greater than UB (upper bound for the trim loss). u_j can equals 1, only if the remainder is greater than UB. Stock length that is longer than UB does not necessarily count as trim loss.
δ_j indicates the remainder of the stock length j.
t_j indicates the extent of trim loss relating to stock length j. $t_j=\delta_j$ for all used stock lengths, except for one that is longer than UB and can be returned to the stock and used again later (where u_j=1) and all unused stock lengths (where y_j=1).

Two cases are possible:

Case 1: the order can be fulfilled as the abundance of material is in stock.

(1) $\min \sum_{j=1}^{m} t_j$ (minimize trim loss which is smaller than UB)

s.t.

(2) $\sum_{i=1}^{n} s_i x_{ij} + \delta_j = d_j (1-y_j) \quad \forall j$ (knapsack constraints, the total length of pieces cut from a stock lengths cannot be longer than the total length of the stock)

(3) $\sum_{j=1}^{m} x_{ij} = b_i \quad \forall i$ (demand constraints, the numbers of pieces are all fixed)

(4) $UB - \delta_j + UB \cdot (u_j - 1) \le 0 \quad \forall j$ (u_j can be 1, only if the remainder of stock length is longer than UB)

(5) $\sum_{j=1}^{m} u_j \le 1$ (maximum number of used stock lengths that do not count as trim loss)

(6) $\delta_j - t_j - (u_j + y_j) \cdot (\max d_j) \le 0 \quad \forall j$ (t_j equals δ_j for all stock lengths where $u_j = t_j = 0$; otherwise t_j can be 0)

(7) $UB \le \max s_i$
$x_{ij} \ge 0$, integer $\quad \forall i, j$
$t_j \ge 0 \quad \forall j$
$\delta_j \ge 0 \quad \forall j$
$u_j \in \{0,1\}$
$y_j \in \{0,1\}$.

Case 2: the order cannot be fulfilled entirely due to shortage of material (the distribution of uncut order lengths is not important).

(1) $\min \sum_{i=1}^{n} \delta_j$ (minimize sum of trim losses)

s.t.

(2) $\sum_{i=1}^{n} s_i x_{ij} + \delta_j = d_j \quad \forall j$ (knapsack constraints)

(3) $\sum_{j=1}^{m} x_{ij} \le b_i \quad \forall i$ (demand constraints)

(4) $x_{ij} \geq 0$, integer $\forall\ i, j$

$\delta_j \geq 0$ $\forall\ j$.

Unutilized stock length that is larger than some *UB* can be used further and is not considered as waste. The question is how to determine *UB*. This depends on the relation between available material and total needs.

Let's consider the case 1 first. If sufficient stock lengths are available there will be cutting plans with "no trim loss" but ever growing stocks. To prevent this an additional condition (case 1, condition (5)) has to be set: only one residual length may be longer than the *UB*. *UB* can be set arbitrarily between 0 and max s_i. The larger *UB* means greater the cutting problem and higher trim loss. *UB* = min s_i is found in practice [9].

In case 2, however, *UB* is not included in the model. If, for example, *UB* is reduced to min s_i, this would lead to the following problem: As the aim of the algorithm is the minimization of overall trim loss, this could lead to unfulfilled requirements for the longest order lengths, even if the overall trim loss would be small and the aim would be achieved according to the logic of the algorithm. The trim losses, which would be longer than *UB* but shorter than the longest order lengths, could remain unutilized. For that reason *UB* shouldn't be less than max s_i. On the other hand if the *UB* would be set to max s_i any trim loss longer than max s_i can certainly be used to cut an additional order length, so *UB* equal or longer than max s_i would not have any influence on the solution. Therefore UB is not included in the 2nd model.

3 Results

For all calculations *MPL/CPLEX* solver on the PC (AMD, 1300 MHz) was used. The data was generated and saved in MS Excel and the solver was called with Visual Basic for Applications. In first experiment we found a solution of a problem described in [10]. The improved solution (Fig. 1) was obtained in 10.1 seconds after examining 59467 integer nodes. The total trim loss is 1 cm, while the solution in [10] has a trim loss of 2 cm. In Fig. 1 firstly all details about order and stock lengths are shown (length and number of pieces demanded for each order length). Then the detailed cutting plan is presented (which and how many order lengths are cut from each stock length) and lastly the trim loss for each stock length and number of realized and unrealized order lengths.

The presented problem is relatively small (4 stock and 5 order lengths) and therefore appropriate for exact method. However with the growing number of integer variables the complexity of the problem and the solution time grow quickly. Sometimes it is not possible to find the optimal solution within reasonable time limit. Fortunately B&B works in such a way that it approaches gradually to the optimal solution and in the mean time offers temporary results, which can be near to optimum. To test the correlation between time limit and trim loss each problem instance in following experiments was solved with 6 different time limits.

For generation of problem instances we decided to use problem generator *PGEN* [12,13]. With *PGEN* it is possible to regenerate test data using the same seed, then to find the solution with some other method and compare the results. Input data are generated according to problem descriptors as random sample of one or more test problems. Problem descriptors are:

n - number of different order lengths
v_1, v_2 - lower and upper bound for order lengths, i.e. $v_1 \leq s_i \leq v_2$ ($i = 1,...,n$)
d - average demand per order length
m - number of non-standard stock lengths
u_1, u_2 - lower and upper bound for non-standard stock length, i.e. $u_1 \leq d_j \leq u_2$ ($j = 1,...,m$).
r - number of consecutive generated problem instances.

Test problems have been generated with the following values of parameters:

- determination of order lengths and demands:

By assigning different values to the problem parameters n (n = 5, 10, 15), v_1 and v_2 (v_1 = 100 and v_2 = 200, v_1 = 200 and v_2 = 400, v_1 = 300 and v_2 = 600) and d (d = 5, 10, 15) and combining them with each other 27 problems have been generated.

- determination of non-standard stock lengths:

Number of non-standard stock lengths m varies from 5 to 15, lower bound u_1 from 1000 to 3000 and upper bound u_2 from 2000 to 6000.

The details about generation of problem descriptors and determination of seed for sequences of test problems are shown in the dynamic programming scheme of the procedure *PROGEN*.

Procedure PROGEN:

for $i = 1$ ***to*** 3
 for $j = 1$ ***to*** 3
  for $k = 1$ ***to*** 3
   $n \leftarrow i \cdot 5$
   $v_1 \leftarrow j \cdot 100$
   $v_2 \leftarrow j \cdot 200$
   $d \leftarrow k \cdot 5$
   $m \leftarrow k \cdot 5$
   $u_1 \leftarrow k \cdot 1000$
   $u_2 \leftarrow k \cdot 2000$
   $c \leftarrow int(\frac{m}{10}) \cdot 9 + 1$
   $seed \leftarrow m + 10 \cdot c \cdot d + 10 \cdot c^2 \cdot v_2 + 1000 \cdot c^2 \cdot v_1 + 1000000 \cdot n$
   $r \leftarrow 10$
   call *PGEN* (n, v_1, v_2, d, m, u_1, u_2, $seed$, r)
  next k
 next j
next i

PROGEN procedure is implemented with Visual Basic. Problem descriptors generated with *PROGEN* procedure for 27 test cases are presented in Table 1. In the table lower and upper bounds for stock and order

lengts are shown, as well as the number of different order and stock lengths and the average demand per order length. The seed for generation of test problems which enables everyone to generate the same test problems is also shown.

For each test case procedure *PGEN* generates 10 consecutive problem instances (r=10). In total there are 270 problem instances, 150 with abundance and 120 with shortage of material.

Each of the 270 problem instances was solved 6 times (using the appropriate model either for lack or abundance of material) with different time limits (time limits were set at 2, 10, 20, 30, 45 and 60 seconds)

The generated data and the solution for first generated instance of the first case are presented in Fig. 2. The optimal solution with trim loss 1 cm was found in 3.9 seconds. The meaning of columns is the same as in Fig. 1.

Stock length 2 is not used in cutting plan and stock length 1 is not cut to the end. Since δ_1=989 and *UB*=102, t_1=0. So stock length 1, which is larger than *UB*, can be used later and is not considered a waste.

The summarized results for all 270 instances with *UB* = min s_i with different time limits are presented in Table 2. For each of the different time limits the total trim loss, percent of trim loss and the number of optimally solved instances is shown. The 3rd column indicates how many instances were solved optimally within the given time limit. In each row 10 problem instances are summarized. Trim loss is calculated as the sum of trim losses of all 10 instances, the percentage is calculated as the average of 10 percentage losses.

In 2 seconds an optimal solution for 57 cases was found, in 60 seconds for 110 cases. The average trim loss varies from 0% to 2.4%.

The trim loss is the largest in case number 7 although the solution is optimal in all 10 instances. Low values of n, d, m mean that in this case the problems are relatively easy to solve, however due to small ratio between stock lengths and order lengths as well as the small number of possible combinations, the optimal solution has a relatively high trim loss. To a lesser extent the same is also true for case No. 4. In other cases the trim loss is 1% or lower. Even better results could be obtained by increasing the time limit for the solution.

4 Comparison with CUT procedure

We have compared results of proposed exact method with the results of SHP CUT described in [10]. The results are shown in Table 3. As in previous table 10 instances are summarized in one line. *UB* is set to min s_i. In the second column it is indicated whether there is enough material or not. Y/N means that in some problem instances there is enough material, in others not. The total trim loss and percent of trim loss with both methods (exact and heuristic method CUT) are shown for each case.

In total within the given time limit (60 seconds) the exact method found a better solution in 64 instances, while the CUT procedure in 139 cases, in 67 cases both methods found the solution with the same trim loss (not necessarily the same solution though). The exact method has better results for cases with smaller d and m (in all cases with d<=5 and m<=5 a better solution was found with the exact method even with small time limits), the CUT procedure for larger cases (d>=10 and m>=10).

5 Selection of the method

Although it is clear from table 3 that exact method is more suitable for smaller cases and heuristics for larger, we need more precise criteria for the selection of solution method in each individual case. The main question that needs to be answered is, what is the maximum size of the problem that can be solved optimally within the given time limit. The question can be answered by using mathematical analysis of computational complexity. But for precise answer we would need a very precise data of speed of particular processor executing specific instructions generated by specific compiler and detailed data about solver. This data is usually not available. Even if they would be, the mathematical analysis would be extremely complicated. Therefore we decided to answer the question by using statistics. The new approach based on the creation of decision tree and its use for the selection of the right method is presented in this chapter.

The main idea of our approach is to generate a large number of cases with different parameters by using the problem generator and then solve them with the selected method and the given time limit. The time limit can be chosen arbitrarily and depends on what is considered as a maximum acceptable solution time in some specific practical situation.

Each case is than assigned either a class value 1 (if optimal solution was found within selected time limit) or 0 (otherwise). Those parameters and class values are then used as the data for decision tree classifiers. Decision trees were chosen as our kind of the problem fulfills the key requirements that are needed for successful implementation of decision trees (as listed by Quinlan [17]):

- attribute-value description: each test case in our example can be described with the same attributes (number of stock and order length, average demand per order length etc.),
- predefined classes: each case is assigned to one of the two predefined classes (either the case can be solved optimally within time limit or not),
- discrete classes: both classes in our example are discrete,
- sufficient data: sufficient number of problem instances can be automatically generated and solved using problem generator and solving procedure,

- "logical" classification models: our example can be expressed as decision trees or sets of production rules.

This approach can be used in various cases:
- to determine whether a certain problem should be solved optimally or heuristically. An example is shown in detail in this section,
- to determine which factors have the greatest influence on time complexity of the problem for the proposed solution method,
- to determine the appropriate size of the sub problem. Sometimes it is possible to either divide a problem in a set of smaller sub problems and solve each sub problem optimally or to solve the majority of the problem heuristically and only the small part optimally (the part that is the most important or can cause higher loss). In order to use his kind of a method, an appropriate size of the sub problem can be determined with this approach. The example of the latter is shown in [15].

As stated earlier we have decided to test the cases with number of stock and order lengths between 5 and 10. The following procedure was used to determine 243 test cases. 5 test instances were generated for each case so we had 1215 problem instance in total:

```
for g=1 to 3
 for h=1 to 3
  for i=1 to 3
   for j=1 to 3
    for k=1 to 3
       u1 ← 1000 . g
       u2 ← 2000 . g
       m ← (j . 2)+3
       d ← (i . 2)+3
       v1 ← h . 100
       v2 ← h . 200
       n ← (g . 2)+3
       seed ← 1000000 .n+1000 .v1+10 .v2+10 .d+m
       r ← 5
       call PGEN (n, v1, v2, d, m, u1, u2, seed, r)
    next k
   next j
  next i
 next g
next h
```

The meaning of the variables is the same as in the previous example.

Each problem instance was then solved with the MPL/CPLEX solver and for every instance the solution time, total trim loss and the fact whether the problem instance was solved optimally or not was recorded. All cases were then distributed into two classes: 1 (optimally solved cases) and 0 (cases not solved optimally).

The whole experiment, which means generating the data and solving all problem instances within the time limit of 1 minute, took just over 10 hours. MS Excel was used for collecting and saving the results. The procedure for the whole experiment was written in Visual Basic for Application. The first 150 problem instances (5 problem instances are summarized in one line) and their solutions are shown in Table 4. The meaning of the variables is the same as in Table 1.

These 1215 cases were then used as the inputs for building a decision tree. First we had to decide which variables to use as attributes. Obviously the variables that are expected to have the influence on the computational complexity of the model should be used. However the number of variables and constraints alone is not a sufficient indicator of time complexity of the problem. Therefore we have chosen the following variables:
- m, n, d - obviously those variables have the influence on the size of the model as m and n influence the number of variables and constraints in the model, while d influence the number of possible combinations.
- v_1, v_2, u_1, u_2 were not included as absolute values but as part of the following ratios:
- r - the ratio between the average stock length and average order length $(u_1+u_2)/(v_1+v_2)$. Earlier it was statistically established that higher ratio means better solutions with heuristic method [10], however the influence of this ratio on exact solution method was not yet studied,
- q - the ratio between the available material and total needs. Problems with higher q should be easier to solve than those with this ratio closer to 1.

70% of the data was used as training, 30% as test data. To avoid over fitting of the data the test required two branches with at least 10 cases. The decision tree shown in Fig. 3 was generated using C5 program. The numbers in the brackets mean how many of the training cases belong to this leaf. The first number is the number of correctly and the second of incorrectly classified cases.

For example: the problems where the ratio between available material and total needs is greater than 2.18239 and number of stock lengths is less or equal to 7 the problem should be solved with exact method. Out of 128 test problems, that fulfill those conditions, a better solution with exact method was found in 126 cases. Other conditions can be explained similarly.

From Fig. 3 it is obvious that for this sort and size of the problem q has the greatest influence on the complexity of the problem, followed by n. On the other hand the influence of the number of order lengths and average demand per order length is surprisingly low. That finding was also used in the selection of sub problem for the C-CUT algorithm [15] where only number of stock lengths and partly the ratio between total material and total needs are pre-set for all sub problems while the number of order lengths and average demand per order length are determined on a case by case basis.

To avoid over fitting of the data the decision tree was tested on the problems mentioned in the first part of the paper. The comparison of the results between CUT and exact method was shown in Table 3. Obviously it would be possible to solve each problem with both methods and keep the best result. However that would require additional time and effort. On the other hand it is possible to solve each problem just once with the method chosen with decision tree.

Using decision tree we've got interesting results. The following cases were solved with the exact method: 1,4,7,11,12,13,16,19,21,22 and 25 while others were solved heuristically. The right decision was made in 21 out of 25 cases (2 cases have the same results with both methods). From the 4 mistakes 3 were resulted in only a marginally higher trim loss. In case 21 the trim loss was considerably higher (438 cm instead of 0).

The most important result is that the total trim loss would be 5593 cm if we would solve all problems with CUT, 32268 cm with exact method and 5310 cm if we solve each problem with the method chosen by the decision tree. This shows that proposed approach can indeed lead to improved results, compared to the results acquired with just one of the available methods.

The other advantage of proposed approach is that it takes both the processing power of the computer and the quality of the solver into account. Obviously the exact method would be selected more often if the experiments would be carried out on considerably faster computer or with better integer programming solver.

6. Conclusion

The article firstly examines the exact solution of G1D-CSP in cases with surplus and lack of material. B&B method and a problem generator *PGEN* for generation of G1D-CSP instances were used.

Three experiments are presented. In the first a better solution for previously published problem is shown. In the second the proposed method was tested by solving 270 problem instances. In the third the comparison with SHP CUT was made. We find out that our method gives better solution for smaller problems, the CUT procedure for larger. The proposed method also approximately shows how close the solutions are to the optimum, while CUT gives no such indication.

The new approach based on decision trees is introduced in order to establish which cutting method should be used. Using a decision tree an appropriate method can be chosen for each individual case based on its size and the probability that the problem of this size can be solved optimally within the given time limit.

The practical implementation of the approach was shown on the example of G1D-CSP, however it can be applied on other types of cutting problems as well. Numerous examples are calculated. The results show a high degree of certainty that the chosen method is the best for specific problem, which reflects in lower trim loss.

Remark. The problem generator PGEN may be obtained from the authors of this article both in source-code and as subroutine executable under Windows. The source code (in VBA) for all experiments, presented in this paper, is also available.

References

[1] Antonio J., Chauvet F., Chu C., Proth J., The cutting stock problem with mixed objectives: Two heuristics based on dynamic programming, European Journal of Operational Research 114 (1999) 395-402.

[2] Bischoff E. E, Wascher G., Cutting and Packing, European Journal of Operational Research 84 (1995) 503-505.

[3] Dyckhoff H., A typology of cutting and packing problems, European Journal of Operational Research 44 (1990) 145-159.

[4] Ferreira J. S., Neves M. A. and Fonseca P., A two-phase roll cutting problem, European Journal of Operational Research 44 (1990) 185-196.

[5] Gau T., Wascher G., CUTGEN1: A problem generator for the Standard One-dimensional Cutting Stock Problem, European Journal of Operational Research 84 (1995) 572-579.

[6] Gilmore P. C. and Gomory R. E., A linear programming approach to the cutting stock problem, Operations Research 9 (1961) 849-859.

[7] Gilmore P. C. and Gomory R. E., A linear programming approach to the cutting stock problem, Part II, Operations Research 11 (1963) 863-888.

[8] Goulimis C., Optimal solutions for the cutting stock problem European Journal of Operational Research 44 (1990) 197-208.

[9] Gradišar M., Jesenko J., Resinovič G., Optimization of roll cutting in clothing industry, Computer & Operation Research 24 (1997) 945-953.

[10] Gradišar M., Resinovič G., Jesenko J., Kljajić M.: A sequential heuristic procedure for one-dimensional cutting, European Journal of Operational Research 114 (1999) 557-68.

[11] Gradišar M., Kljajić M., Resinovič G., A hybrid approach for optimization of one-dimensional cutting, European Journal of Operational Research 119 (1999) 165-174.

[12] Gradišar M., Resinovič G., Kljajić M., Evaluation of algorithms for one-dimensional cutting, Working paper No. 90, Faculty of economics, University of Ljubljana, 1999.

[13] Gradišar M., Resinovič G., Kljajić M., Evaluation of algorithms for one-dimensional cutting, Computers & Operations Research 29 (2002) 1207-1220.

[14] Gradišar M, Trkman P., Indihar Štemberger M.: Exact solution of general one-dimensional cutting stock problem. Working paper No. 123, Faculty of Economics, University of Ljubljana, 2001

[15] Gradišar M., Trkman P.: A combined approach for solution of general one-dimensional cutting stock problem. Working paper No. 124, Faculty of Economics, University of Ljubljana, 2002.

[16] Haessler R. W., Vonderembse M. A., A procedure for solving the master slab cutting stock problem in the steel industry, AIIE Trans 11 (1979) 160-165.

[17] Quinlan J. R.: C 4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.

[18] Schilling G., Georgiadis M., C., An Algorithm for the Determination of Optimal Cutting Patterns, Computers & Operations Research 29 (2002) 1041-1058.

[19] Stadtler H., A one-dimensional cutting stock problem in the aluminium industry and its solution, European Journal of Operational Research 44 (1990) 209-23.

[20] Sweeney P. E. and Paternoster E. R., Cutting and packing problems: A Categorised, Application-Orientated Research Bibliography, Journal of the Operational Research Society 43 (1992) 691-706.

[21] Vance P.H., Branch-and-Price Algorithms for the One-Dimensional Cutting Stock Problem,Computational optimization and applications 9 (1998) 211-28.

[22] Vanderbeck F., Computational Study of a Column Generation Algorithm for Bin Packing and Cutting Stock Problems, Research Papers in Management Studies, No 14, University of Cambridge, 1996.

[23] Wascher G., Gau T., Generating Almost Optimal Solutions for the Integer One-dimensional Cutting Stock Problem, Working Paper No. 94/06, Institut fur Wirtschaftswissenshaften, Technische Universitat Braunschweig, 1994.

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research institution in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 700 staff, has 500 researchers, about 250 of whom are postgraduates, over 200 of whom have doctorates (Ph.D.), and around 150 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of S**love**nia (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

In the last year on the site of the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

At the present time, part of the Institute is being reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project is being developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park will take the form of a shareholding company and will host an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of Economic Relations and Development, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel.:+386 1 4773 900, Fax.:+386 1 219 385
Tlx.:31 296 JOSTIN SI
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Contact person for the Park: Iztok Lesjak, M.Sc.
Public relations: Natalija Polenec

CONTENTS OF *Informatica* Volume 27 (2003) pp. 1–503

Papers

Editorials

Professional Societies

INFORMATICA

AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS

INVITATION, COOPERATION

Submissions and Refereeing

Please submit three copies of the manuscript with good copies of the figures and photographs to one of the editors from the Editorial Board or to the Contact Person. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible directly on the manuscript, from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks. If the paper is accepted, the editor will also send copies to the Contact Person. The Executive Board will inform the author that the paper has been accepted, in which case it will be published within one year of receipt of e-mails with the text in Informatica LaTeX format and figures in `.eps` format. The original figures can also be sent on separate sheets. Style and examples of papers can be obtained by e-mail from the Contact Person or from FTP or WWW (see the last page of Informatica).

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Drago Torkar, Informatica, Institut Jožef Stefan, Jamova 39, 1111 Ljubljana, Slovenia.

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than ten years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering the European computer science and informatics community - scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name: .....

Title and Profession (optional): .....

.....

Home Address and Telephone (optional): .....

.....

Office Address and Telephone (optional): .....

.....

E-mail Address (optional): .....

Signature and Date: .....

Informatica WWW:

http://ai.ijs.si/informatica/
http://orca.st.usm.edu/informatica/

Referees:

Informatica

An International Journal of Computing and Informatics

Archive of abstracts may be accessed at USA: http://, Europe: http://ai.ijs.si/informatica, Asia: http://www.comp.nus.edu.sg/ liuh/Informatica/index.html.

Subscription Information Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 1000 Ljubljana, Slovenia.
The subscription rate for 2003 (Volume 27) is
– USD 80 for institutions,
– USD 40 for individuals, and
– USD 20 for students
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

LaTeX Tech. Support: Borut Žnidar, Kranj, Slovenia.
Lectorship: Fergus F. Smith, AMIDAS d.o.o., Cankarjevo nabrežje 11, Ljubljana, Slovenia.
Printed by Biro M, d.o.o., Žibertova 1, 1000 Ljubljana, Slovenia.

Orders for subscription may be placed by telephone or fax using any major credit card. Please call Mr. R. Murn, Jožef Stefan Institute: Tel (+386) 1 4773 900, Fax (+386) 1 219 385, or send checks or VISA card number or use the bank account number 900–27620–5159/4 Nova Ljubljanska Banka d.d. Slovenia (LB 50101-678-51841 for domestic subscribers only).

Informatica is published in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Franjo Pernuš)
Slovenian Artificial Intelligence Society; Cognitive Science Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)
Automatic Control Society of Slovenia (Borut Zupančič)
Slovenian Association of Technical and Natural Sciences / Engineering Academy of Slovenia (Igor Grabec)
ACM Slovenia (Dunja Mladenič)

Informatica is surveyed by: AI and Robotic Abstracts, AI References, ACM Computing Surveys, ACM Digital Library, Applied Science & Techn. Index, COMPENDEX*PLUS, Computer ASAP, Computer Literature Index, Cur. Cont. & Comp. & Math. Sear., Current Mathematical Publications, Cybernetica Newsletter, DBLP Computer Science Bibliography, Engineering Index, INSPEC, Linguistics and Language Behaviour Abstracts, Mathematical Reviews, MathSci, Sociological Abstracts, Uncover, Zentralblatt für Mathematik

The issuing of the Informatica journal is financially supported by the Ministry of Education, Science and Sport, Trg OF 13, 1000 Ljubljana, Slovenia.

Volume 27 Number 4 December 2003 **ISSN 0350-5596**

Informatica

An International Journal of Computing and Informatics