INFORMATICA 2/89

AN ADAPTABLE PARALLEL SEARCH OF KNOWLEDGE BASES WITH BEAM SEARCH*

Keywords: expert system, knowledge base, beam search, parallel search

S. Prešern, P. Brajak, L. Vogel and A. P. Železnikar
Iskra Delta, Ljubljana

This paper describes an on-line computer supported expert system, which is designed for searching complex knowledge bases. Searching is performed in parallel with beam search, so that only a limited number of specific features are examined on each level. Parameters for beam search are modified during application of the software package, so that self adapting capability of the beam search through an expert system is incorporated in the design of a package. The proposed approach is especially effective for image processing and speech recognition. The difference between making a sequential program parallel and using natural parallelism is shown. The proposed method on a mesh network of Tränsputers or on a MIMD parallel computer PARSYS has properties of a neural network.

1. INTRODUCTION

The need for more powerful computers is growing faster than technology. Parallel processing is a very powerful solution to many processing-intensive applications as for example image processing, numerical problems, artificial intelligence, speech recognition and others. Even though many parallel computers are already on the market, most software approaches are based on von Neuman logic. Many sequential algorithms are simply parallelised for a parallel environment, which means not using all the capabilities and approaches of parallelism. Other different solutions have also been proposed, as for example a neural network for visual pattern recognition (3), which has also a self-organizating capability of the network, the neural network processor for speech recognition (5), and others.

Instead of rewriting sequential programs to a parallel form, we suggest approaching the problem in a different way. The natural parallelism of the problem has to be explored and used directly in a parallel computer system. In natural parallelism, a problem is specified by a set of objects and relations between those objects. The pool of processing elements performs the task in a self organized fashion so that available processors perform the task. Using an expert system in the background and enabling adaptability of the system to the environment, we propose a new approach to complex application software approach.

An expert system is assumed as a support in our parallel computer system applications for a parallel search of a knowledge base. Most complex applications should interact with the changing environment, which means a need or capability to react and adapt to those changes. The role of an expert system is to enable easy man-machine interfacing and to enable adaptability to the changing environment. Such adaptability is called genetic adaptability.

2. KNOWLEDGE BASE

The knowledge base is the fundamental framework for the description of the domain of the problem. It represents the core of the system and contains all relevant domain-specific information, permitting the system to behave as an intelligent specialist. A domain might be a set of industrial parts or objects (9) (typically for automatic recognition of industrial objects in robotics), analysis of different properties by testing (printed board testing, chemical testing) or socio-economic properties of interacting subjects (in behavior simulation, macroeconomic model simulation) etc. The knowledge base changes its content and connectivity as new information is added to the system. Therefore we have a dynamic system for performing cognitive tasks. Information processing in dynamic systems is studied by Harmony theory (11).

2.1. Elements of a Knowledge Base

In order to represent a knowledge base we must have a symbolic description of the properties of the objects which are connected by relations.

A domain oriented network consists of n objects. Each object o(i) forms a knowledge atom in a knowledge network and is represented by a vector of m properties P(i,j)

$$o(i) == P(i,j) \qquad 1 >= i >= n, \quad 1 >= j >= m, \quad m =< n$$

All objects form a P matrix of properties

$$P = \begin{matrix} P(1,1) & P(1,2) & \dots & P(1,m) \\ P(2,1) & P(2,2) & \dots & P(2,m) \\ \vdots & \vdots & \dots & \\ P(n,1) & P(n,2) & \dots & P(n,m) \end{matrix}$$

(*) This paper was originally presented at the 22-nd Annual Hawaii International Conference on System Sciences, Hawaii, Jan. 1989

Recognition includes performing a set of measurements and feature extractions on an unknown object in order to analyze some properties of the object. The identification process consists of a set of feature extractions F. A set of feature extractions F consists of h feature extractions:

$$F = (f(1), f(2),\ldots,f(j),f(j+1),\ldots f(h))$$

The result we get after performing a single feature extraction f(j), is one of the previously stored values. This is the representational feature V(j,i) of the knowledge atom. We say that a feature extraction f(j) consists of k elements V(j,i)

$$f(j) = (V(j,1),V(j,2),\ldots V(j,i),\ldots V(j,k))$$

The probability of obtaining the value V(j,i) by feature extraction f(j) equals p(j,i). All probabilities p(j,i) (1=<i=<n) for realization of an event A(i) by feature extraction f(j) form a set P(j)

$$P(j) = (p(j,1),p(j,2),\ldots p(j,i),\ldots p(j,k))$$

with a property

$$\sum p(j,i) = 1 \qquad 1 >= i >= n$$

A different feature extraction f(j+1) consists of g different elements V(j+1,i)

$$f(j+1) = (V(j+1,1),\ldots V(j+1,i),\ldots V(j+1,g))$$

with probabilities p(j+1,i) for realization of a representational feature V(j+1,i)

$$P(j+1) = (p(j+1,1),\ldots p(j+1,i),\ldots p(j+1,g))$$

All objects o(i) (1=<i=<n) which are treated by a computer supported sensing system form a set N

$$N = (o(1),o(2),\ldots,o(n))$$

The representational features of all objects o(i) are represented in a matrix form

	f(1)... f(h)
o(1)	V(1,1)...V(h,1)
o(2)	V(1,2)...V(h,2)
.	
.	
o(n)	V(1,n)...V(h,n)

where each row represents a symbolic description of an object o(i) and each column represents possible outcomes of each feature extraction f(i).

The problem of object recognition is solved when an unknown object o(i) is identified as one element of the set N.

The distinction between levels of description must be clear: a microlevel involving knowledge atoms, and a macrolevel involving individual features by themselves. The relationship between those two levels is very important. The knowledge atoms are determined by the nature of the objects and form the problem domain. On the other hand, the automatic cognition process always detects only a single representational feature.

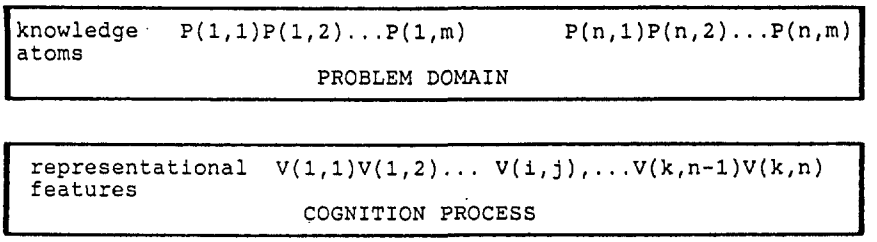

Fig.1. Knowledge atoms form a problem domain and representational features are detected in cognition process

All representational features V(i,1), V(i,2),... represent the cognitive system's representation of possible states of the environment with which it deals. In the environment of object recognition, these features are for example individual pixels, edges, syntactic description of an image, holes and identification of objects. In medical diagnoses features are symptoms, outcomes of tests, diseases, prognosis and treatments. In circuit analysis the features are high current through some resistor, high voltage on certain pin, or low voltage at a transistor.

A knowledge network also requires connections between knowledge atoms and representational features. These connections can form a single a or multilevel network.

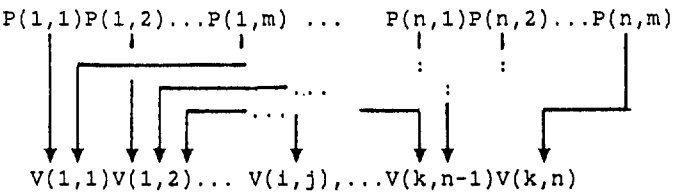

Fig.2. Configuration of a single level network

Symbolic representation of each knowledge atom is simply a value vector V of 1 and 0 values which denote on and off connection between the corresponding nodes.

```
V(O(1)) => (0  0  1        1  0       0  0)
V(O(2)) => (0  1  0        1  0       0  0)
    .
    .
V(O(n)) => (0  0  0        1  0       1  0)
```

The first index in a representational feature means the type of feature extraction and the second index is a value of a particular feature extraction.

2.2. Construction of the Knowledge Network

The upper level of a knowledge base is a set of production rules determining connections between atoms and representational features together with atom definitions. Each rule consists of a precondition and an action. The

precondition contains Boolean combinations of clauses, each of which applies a predicate to an object in the system and tests its value. The action of a rule gives a new description of a representational state that can be drawn if the precondition is satisfied.

Knowledge source:

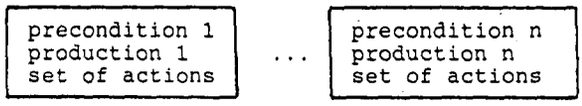

Fig.3. Knowledge source consisting of preconditions, productions and actions

A representational feature has the role of a pattern, which is a precondition to the production (Fig.3). Each production is a

PRODUCTION RULE
- update the set of possible objects
- update the set of possible productions
- choose the next cheapest action
- perform the choosen action.

Let us form a network for knowledge representation. On the highest level there is the whole set of possible representational features V. After the first feature extraction is performed, the possible representational features are:

1st feature extraction:
V(1,1), V(2,1),...V(s,1)

Some of those representational features have the same value and most knowledge atoms can not be distinguished one form another. Therefore a sequence of feature extraction has to be performed in order to find a knowledge atom and identify an object.

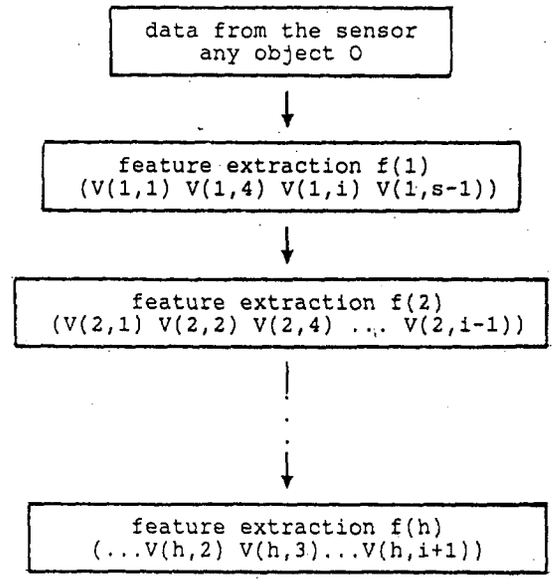

Fig.4: A sequence of feature extractions in a sequential program

selection of appropriate routing, based on present and past representational states. The set of actions includes pruning of a search tree and update of the representational state.

The use of pattern based knowledge lends itself to production rules. Whenever a pattern is matched, a corresponding action is taken which enables the sensing-based recognition system to act on the information obtained by matching the pattern. Each pattern becomes the condition of a production rule and is associated with a certain action. These patterns have to be matched in parallel. A pattern is data which are obtained by feature extraction. Therefore, a set of patterns for each object is actually a symbolic description of the object's features.

The general structure of the knowledge source which is used by the planner to identify an object, is that sensor data are obtained and that the pattern is used to choose the production rule. Therefore a planner consists of a pattern and a production rule:

PRODUCTION
- get a pattern

The parentheses below each measurement state the possible representational features. In a sequential activation of a particular feature extractor this sequence of feature extractions forms a knowledge representation which is the basis for a search tree.

The spanning of the knowledge tree on this level is equivalent to the number of different values of representational features (Fig.5).

The knowledge atoms are placed on the lowest level of the knowledge network.

This organization is appropriate for a nonparallel approach. The recognition is performed by comparison between the vector of properties P(i) defining the knowledge atom and the representational feature vector V(O(i)). In our task of cognition we perform pruning in stages. The goal is reached when a particular representational feature has an active connection to only one knowledge atom. This is a part of the decision-making process in Harmony theory.

The knowledge network is organized as a collection of hierarchical descriptions where each level in the hierarchy represents a

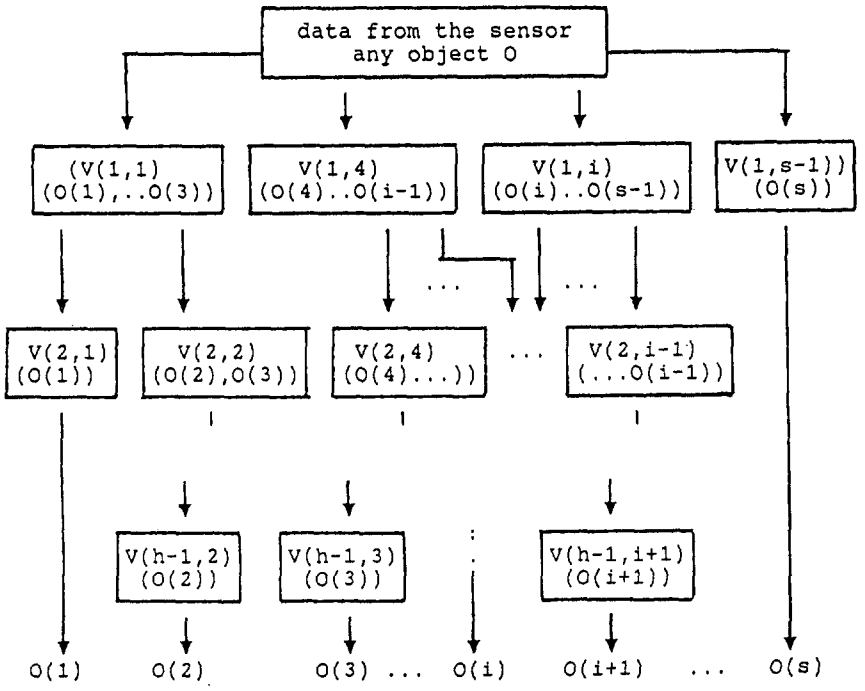

Fig.5. Knowledge representation for object recognition

different conceptual abstraction of the object symbolic representation. Spanning of the knowledge network is not a problem if parallel processing on a parallel computer with shared memory is proposed. If a mesh architecture with message passing is used, then a mapping of the knowledge network to the mesh architecture has to be done.

Analysis of different properties and relations in the knowledge network requires a different complexity and therefore a different amount of computer time. The strategy or sequence of operations is very important. The strategy is determined by the knowledge network organization. Therefore, the final step in knowledge network construction is optimization, so that the shortest average time is required for object recognition.

3. NATURAL PARTITIONING AND MAPPING TO THE PARALLEL COMPUTER

The process of object recognition requires an adaptable algorithm which has to be mapped to a mesh network of Transputers. The mapping is required when the communication graph of a parallel algorithm differs from the interconnection architecture of the physical parallel machine. Several solutions have been proposed for mapping algorithms to CHiP machine (2) array processors (6), Star computers (7) and others.

For a given harmony model, every node in the network is mapped to one processor, and every link in the network becomes a communication link between two processors. The processors each have two possible messages with values for the representational feature processors; 1 = active and 0 = inactive. The completion is denoted by the code identity number at any processor.

The knowledge network consists of features which are describing objects. Each feature extractor is, in terms of parallel search, a software process. A knowledge network is designed as an interconnected set of processes. Each process is an independent software unit and a feature, forming the world representation. Depending on the result of the present state, a new set of actions is performed. This means that the result on each level guides the search and therefore a self routing is performed in real time. This means that each process communicates with other processes along four channels, limited by the physical design. A logical design is hidden and is specified by the messages between processors.

3.1. Parallelization

In general a knowledge network permits any number of links between different objects which is acceptable for a shared memory parallel computer system. But the Transputer system is a typical message passing parallel computer system limited to four neighboring links and forming a mesh architecture. Therefore a mapping has to be performed in order to:

- limit the number of links to four
- avoid backward links
- reduce the unnecessary links.

In the case of object recognition, processes running in parallel on different Transputers represent feature extraction. Communication is achieved by message passing to communication channels. Massages have to be sent to only those processors with the correct identity number. The code in each Transputer has four sections:

- a feature extractor
- the pruning of a search tree
- a decision which feature to extract next
- the sending of a message to the neighbor to perform the next feature extraction.

In our model a general knowledge is stored in a long term memory. Each level of long term memory contains symbolic descriptions of the physical properties of objects in a representational feature vector. Primitive object elements are organized as levels of different classes. Each feature extractor f(i) is a process on one processor. Depending on a result of feature extraction, the correct branch in a network is selected (Fig.6).

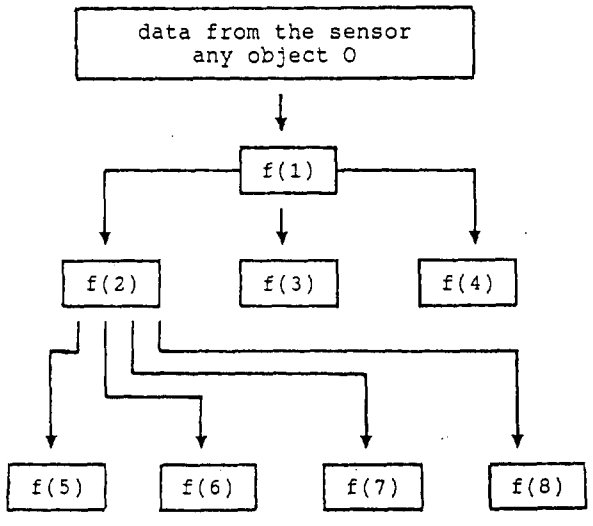

Fig.6. The original tree of feature extractors or processors

The symbolic representation, formed in the knowledge network, is therefore mapped onto the physical network of Transputers. All network interconnections are limited to four physical connecting links. A grammer-based description for generating embeddings of large binary trees in square processor array was suggested by Bailey (1).

In our mapping procedure some feature extractors have to be multiplicated for two reasons (fig.7):

(1) in order to enable enough connections, and
(2) in order to enable connections to dislocated feature extractors.

This transformation process can be done automatically in stages. Fig. 7 shows mapping of the original tree of processes from fig. 6 to the mesh architecture.

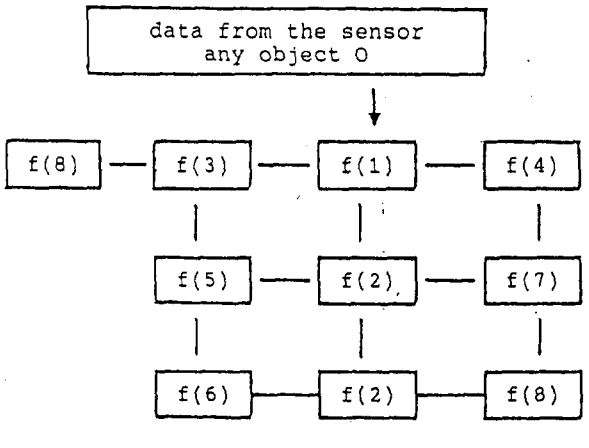

Fig.7. Mapping to the mesh architecture

The algorithms for computing tree functions for systems where a search tree is represented by a list of edges, so that each undirected edge is represented by two directed edges in this list, have been proposed by Gopalakrishnan (4).

The programming approach on Transputers is based on OCCAM language and the problem has to be divided into large grain parallelism. Within a process running on one Transputer even more fine grain parallelism can be achieved by using the opportunity for fast context switch provided by the hardware.

After every stage of feature extraction, the unknown object is better defined than on the previous stage. The process of identification is considered as a multistage recognition, where each feature is processed on a separate Transputer. The object recognition is in that sense similar to a bubble movement in a bubble sort. Some proposals for a bubble sort on an array of Transputers have been already proposed (8).

We have seen a parallelization approach where each processor is dedicated to one process. This approach is not very effective because Transputers are not loaded in balance and because a network can not have genetic properties. Therefore a pool of waiting processes has to be available in each Transputer. An internal processor utilization has to be measured for a large number of runs in order to get an average load balance. This is done with monitor processes or simulation processes to represent parts of the application program.

3.2. Natural Parallelism

Using the natural structure of a problem and the natural parallelization we partition the knowledge network, the feature extraction and search algorithm.

Natural partitioning simplifies the task of system design and programming. The knowledge network is transformed to a hierarchical logical structure, realized on a variable number of processors, depending on the size of a problem. By using natural partitioning, there is no contention for the communication mechanism, regardless of the number of Transputers in the system. The mapping of the problem to the computer hardware is not limited to the number of Transputers, because arbitrary size and topology can be constructed.

Using natural parallelism in a Transputer system, the input data, as for example the image, are broadcasted to the first column of Transputers. The whole first row is performing feature extraction in parallel and pruning is not necessary after each feature extraction. The required condition is that we have as many Transputers as we have processes for feature extraction (Fig.8).

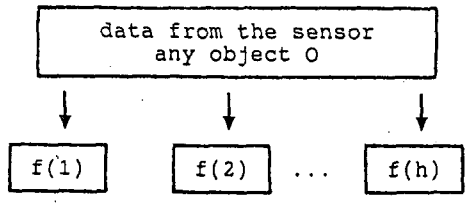

Fig.8. Parallel feature extraction

Each processor performs the feature extraction procedure and gets a vector describing which knowledge atoms have the same representational feature value as in the input. This representational vector is for example on processor 1 (0,1,1,0,0,0,0) which means that knowledge atoms 2 and 3 have corresponding value for the feature 1. The same procedure is performed on other processors (Fig.9).

knowledge atoms: 1 2 3 4 5 6 7		
processor 1:	(0,1,1,0,0,0,0)	F.E.1
processor 2:	(0,1,1,1,1,0,1)	F.E.2
processor 3:	(1,0,1,0,1,0,0)	F.E.3
processor 4:	(0,0,1,0,0,0,0)	F.E.4
AND	(0,0,1,0,0,0,0)	

F.E.# is the feature extraction #

Fig.9.: The representational vectors on 4 processors performing 4 feature extractions.

The unknown object is identified simply by making an AND operation on all representational vectors. A corresponding knowledge atom where an AND operation gives "1" is the identified object.

If the number of processors is smaller than the number of required feature extractors, then each processor has two or more feature extractors. One is in the running state, others are waiting. Comparison is performed after each cycle of feature extraction. Let us look at an example with 2 processors and four feature extractors (Fig. 10).

knowledge atoms: 1 2 3 4 5 6 7		
processor 1:	(0,1,1,0,0,0,0)	F.E.1
processor 2:	(0,1,1,1,1,0,1)	F.E.2
AND	(0,1,1,0,0,0,0)	

PRUNE THE KNOWLEDGE ATOMS

knowledge atoms: 2,3		
processor 1:	(1,0,1,0,1,0,0)	F.E.3
processor 2:	(0,0,1,0,0,0,0)	F.E.4
	(0,1,1,0,0,0,0)	AND P.S.
AND	(0,0,1,0,0,0,0)	

P.S. denotes previous step

Fig.10.: The representational vector on 2 processors performing 4 feature extractions.

The representational state of the cognitive system is determined by an AND logical operation of values for all the representational variables (V(i)), a so called representation vector.

If the number of processors is smaller than the number of processes, then a combination of tree pruning and parallel feature extraction has to be performed.

We see that in the case when the number of processes is bigger than the number of processors, the sequence of activating processors is important. If the sequence is different we might identify the object sooner. The upper example gives the answer already after the first run if features 3 and 4 are executed before the feature extractors 1 and 2. The arrival of particular type of objects is the criteria for determining the optimal sequence of processes execution. The criteria is to require minimal average time for object recognition. Therefore the history of identified objects has to be collected and ocasionaly the sequence of feature extractors has to be adapted if the changing environment requires so. This is a task for the expert system.

The proposed approach is very convenient also because most speed is gained simply by parallel execution of the whole feature extractor procedures and no recoding of algorithms is required.

3.3. Neural Network

Let us sum up the computer environment. We have a parallel machine with distributed processing capability, distributed memory and distributed knowledge. Our problem is a problem of fuzzy reasoning on a symbolic world structure. The system has properties of neural network as for example:

- Patterns (representational features) are not locally memorized but are memorized over the whole system in a sense that properties of an object are stored in different memory modules.

- Object recognition is performed through feature extraction, that means symbolic representation rather than memorizing the whole image. That is exactly the way that living systems work.

- The network is fault tolerant. In that sense, the system behaves like a neural network because if a part of the system is damaged or fails, the system still works, although the probability for wrong identification is higher.

- Recognition and learning are performed by local adaptation in the sense that a set of knowledge atoms is updated, a set of representational features is updated, and a set of distribution probabilities is locally updated. A new strategy is selected in real time.

- The dynamic of the system is parallel and asynchronous.

- The knowledge network contains elements which have a high percentage of fuzziness.

The nature of our expert system is parallelism. The system is composed of a number of modules, executed on different processors. The calling of the module depends on the data environment. The knowledge network including all steps of symbolic representation, normalization and optimization gives the framework for parallelism.

Future applications of parallel processing will not be based on static expert systems but rather on knowledge engineering. This is a shift from mere data processing to an intelligent processing of knowledge.

4. GENETIC PARALLEL SEARCH

The interesting activity in sixth generation computer projects is development of genetic algorithms which adapt to the changing environment.

In the second paragraph we have seen that strategy depends on knowledge network organization. Knowledge network organization depends on the optimization process which is described before. But the optimization process and therefore strategy strongly depends on the test pattern.

Let us see an example: we have a domain for object recognition of 100 industrial objects. Some objects do require very complex analysis. But in the recognition process only a subset of 5 objects appear and they are easily recognized by one typical feature. A conventional static object recognition system would perform always the same search designed for 100 types of knowledge atoms. But more flexible systems for future generation parallel computers will have the capability to learn and adapt the search algorithm according to the environment. Therefore the system has to keep track of the recent history of objects and accordingly adapt the search strategy.

In order to perform algorithm adaptability, each data has to have a weighting function describing its importance for problem solving, as for example object recognition. A method is proposed to acquire this global information by calculating probability factors that any unknown object is recognized fast.

A set of n objects has the distribution a(i) where a(i) is the percentage of objects a(i) in a set of n objects.

$$\sum_i a(i) = 1 \qquad 1 >= i >= n$$

The number of objects i in a set of expected domain equals N, so that

$$a(i) = \frac{N(i)}{\sum_i N(i)}$$

where N(i) is the number of objects i in the whole set of objects. At the same time this means that probability for an unknown object to be object i equals a(i).

For the whole set of n objects we get a probability vector

$$A = (a(1)\ \ a(2),\ \ \ldots\ \ a(n))$$

An adaptable parallel search means a capability of the system to change the search strategy dynamicaly according to the present state and history of the search. Strategy for further search depends on conditions which allow specifications of different plans for different data from the environment. The advantage is that the adaptable search can immediately try the correct plan instead of searching an appropriate plan and backtracking to correct itself.

A genetic computer must have the capability to modify itself by learning. The learning phase is denoted by three steps:

- assign values to representational features of a knowledge atom,
- activate connections to knowledge atoms that are consistent with the representation and
- confirm unique representation of a new knowledge atom

else

- suggest a new feature extraction and enlarge a dimension of representational state.

Present technology gives us an opportunity to perform an adaptable parallel search of knowledge bases on an array of Transputers.

The Transputer is programmed to perform a specialized function and is regarded as a black box thereafter. This specialized function is feature extraction and search information, depending on the present result of the feature extraction. The adaptable parallel search is performed by a network of programmable components which have the mesh topology, limited to four links to each Transputer.

5. CONCLUSION

We have discussed an adaptable parallel search supported by an expert system. This subject is a goal of the sixth generation computers which are based on a man-behavior intelligence or neural intelligence (12). Present research projects in parallel processing design have shown very good results in hardware design using a massively parallel structure with a high interconnectivity between large number of processors.

By speculation and having in mind living organisms we can predict future trends in genetic computing. The parallel system would perform a self balancing statistics so that processor utility is measured, bottlenecks are identified and rescheduling of strategy is performed.

Nowadays pattern directed systems are proposed as future software organization on parallel machines. We feel that even on parallel machine we need a deterministic fixed calling system but upgraded by a genetic capability to adapt, reformulate, reorganize according to performance measurements and to changing environment.

Not many new concepts in problem approach have been proposed with parallel processing. We feel that parallelization of sequential programs is only the first step and the bridge between sequential problem approach and parallel machines. But real parallelism on parallel machines has different problem formulation, partitioning and adaptability. The principles which were proposed in this paper show how to combine the sixth-generation computer project and artificial intelligence and are under development on a PARSYS parallel computer system (10). The approach is general enough even to be performed on any MIMD computer or on a Transputer based parallel machine which was demonstrated by examples.

Having this in mind we have designed a massively parallel computer system PARSYS with up to 64 powerful 32 bit processing units and up to 64 memory modules.

6. REFERENCES

(1) D.A. Bailey and J.E. Cuny, "An Efficient Embedding of Large Trees in Processor Grids", Proc. of the 1986 Int. Conf. on Parallel Processing, IEEE Competer Society, p. 819, 1986.

(2) F. Berman, "PREP-P: A Mapping Preprocessor for CHiP Computers", Proc. of the 1985 Int. Conf. on Parallel Processing, IEEE Computer society, p. 731, 1985.

(3) K. Fukushima, "A Neural Network for Visual Pattern Recognition", IEEE Computer, p. 65, 1988.

(4) P.S. Gopalakrishnan et al., "Computing Tree Functions on Mesh-Connected Computers", Proc. of the 1985 Int. Conf. on Parallel Processing, IEEE Computer society, p. 703, 1985.

(5) T. Kohonen, "The 'Neural' Phonetic Typewriter", Helsinki University of Technology, IEEE Computer, March 1988, p.11, 1988.

(6) T. Lin and D.I. Moldovan, "Tradeoffs in Mapping Algorithms to Array Processors", Proc. of the 1985 Int. Conf. on Parallel Processing, IEEE Computer society, p. 719, 1985.

(7) W. Lin and C. Wu, "Design of Configuration Algorithms for Commonly-used Topologies for a Multiprocessor - STAR", Proc. of the 1985 Int. Conf. on Parallel Processing, IEEE Computer society, p. 734, 1985.

(8) J. Modi and R. Prager, "Implementation of Bubble Sort and the Odd-even Transpozition Sort on a Rack of Transputers", Parallel Computing, Vol.4, No.3, June 1987.

(9) S. Prešern and L. Gyergyek, "An Intelligent Tactile Sensor - An On-line Hierarchical Object and Seam Analyzer", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No.2, March 1983.

(10) S. Prešern et. al., "PARSYS - A Massively Parallel Computer System Project in Yugoslavia", ISMM Int. Conf. on Mini and Microcomputers, Florida, 1988.

(11) D. E. Rumelhart et al., "Parallel Distributed Processing", MIT Press, Cambridge, Massachusetts, 1986.

(12) B. Souček, "Neural and Massively Parallel Computers: The Sixth Generation", John Wiley & Sons, Inc., 1988.