Strojniški vestnik - Journal of Mechanical Engineering 56(2010)5, 330-339
UDC 658.5
Paper received: 18.11.2009
Paper accepted: 04.05.2010

Solving JSSP by Introducing Hamilton Similarity and Time Dependent Fitness Scaling

Arijan Abrashi[1,*] - Nedjeljko Štefanić[2]- Dragutin Lisjak[2]
[1]Energy and Environmental Protection Institute, Croatia
[2]Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, Croatia

In this paper we propose and test a niching genetic algorithm (GA), which uses the so-called Hamilton similarity for a comparison of individuals in the population. The advantage of the Hamilton similarity lies in the fact that there is no need for context sensitive information in order to successfully compare two population members. Furthermore, the algorithm was tested on the famous Job Shop Scheduling Problem (JSSP) - benchmark mt10, and statistical results of the test were given. Significantly smaller standard deviation of the proposed GA compared to Simple GA clearly demonstrates its superiority.

In addition to the Hamilton similarity, time dependent fitness scaling was proposed which in conjunction with niching significantly reduces the probability of the algorithm to get stuck in one of the less desirable local optimum. Finally, suggestions for future research are given.

Keywords: genetic algorithm, niching, Hamilton similarity, time dependent fitness scaling, Job shop scheduling problem

0 INTRODUCTION

Today, genetic algorithms as part of the wider area of artificial intelligence are more frequently used in solving real-life problems. Fifteen years ago, genetic algorithms were known only to a few scientists, professors and their students and their study was limited to the academic community. Today the situation is quite different. Interest in genetic algorithms has been shown by various branches of science; from economics, biology and medicine to computer science, engineering, operations research and social science [1]. Such a broad interest in genetic algorithms is the result of the fact that genetic algorithms are able to successfully tackle NP-hard (nondeterministic polynomial-time hard) problems.

The JSSP discussed in this paper falls into the permutation class of problems that generally speaking brings additional difficulties to genetic algorithms. Because of the permutation nature of the problem, illegal solutions (offspring) are common, and this has a very negative impact on algorithm performance. Despite all the difficulties, first attempts to use genetic algorithms in solving the permutation class of problems appeared more than 20 years ago and four of those attempts were particularly important [2]:

- Traveling salesman problem (Goldberg & Lingle, 1985),
- Scheduling (Davis, 1985),
- Integrated circuit design (Louis & Rawling, 1991),
- Vehicle route planning (Blanton Jr. & Weinwright, 1993).

The present knowledge in the field of genetic algorithm allows significant progress in solving the permutation class of problems. Furthermore, a better understanding of selection mechanism and development of new operators, have placed genetic algorithms among the best and most widely applicable optimization techniques.

1 NICHING GENETIC ALGORITHMS

Niching genetic algorithms differ from simple genetic algorithm in a way that it does not allow one (best) individual to dominate through the entire selection process. This feature is extremely important because it implicitly reduces the ability of the algorithm to get stuck in a local optimum.

[*] Corr. Author's Address: Energy and Environmental Protection Institute, Koranska 5, 10000 Zagreb, Croatia, arijan.abrashi@ekonerg.hr

Although, in the literature different niching methods can be found [3] and [4], the most popular approach by far was proposed by David Goldberg by the name of Sharing [1]. In Goldberg's approach, fitness function value of each individual in the population is reduced in proportion to its similarity to other members of the population. In this way, the ability of individual members to participate in the creation of the next generation is substantially diminished, which indirectly prevents domination of the best member. Such an approach logically implies the question of how to determine a degree of similarity between individual members of the population. There are two approaches, phenotypic and genotypic. The genotypic approach compares individuals on the chromosome level, while the phenotypic approach compares individuals' characteristics, which an algorithm tries to optimize. The method proposed in this paper shows the third approach, the so called Hamilton similarity. The basic condition for using Hamilton similarity is chromosome representation in the form of the not so often used diploid chromosome.

1.1 Diploid vs. Haploid Chromosome

In the area of genetic algorithms, the implementation of diploid chromosome is extremely rare. The majority of research where diploid chromosome is used, focus on solving problems with time-changing fitness function [5]. These studies have shown that genetic algorithm that uses diploid chromosome, due to a greater genetic diversity of the population; react quicker to changes in the environment (time-changing fitness function).

The primary goal of diploid chromosome representation is maintaining the weaker gene or the weaker group of genes in the population that at a certain point in the evolution are not of a great use but could become so in the near future. This principle allows greater retention of genetic diversity in the population and increases the probability of finding the global optimum. The claim that diploid chromosome retains greater genetic diversity was examined and proved by Goldberg and Smith [6] on the problem of "counting ones" in 1987. In its study chromosomes were represented in binary format (ones and zeros) where the ones contributed more to individual fitness values than zeros, therefore, coefficient r is always greater than one.

$$r = \frac{f_1}{f_0} > 1 \,, \tag{1}$$

where f_1 is contribution to the total value of individual by ones, while f_0 is contribution of zeros.

In the case of haploid representation, chromosomes with a large number of zeros slowly disappear from the population. On the other hand, the population composed of individuals with diploid chromosome in a greater extent is able to defend zeros and in this way maintain higher genetic diversity, which is a very desirable characteristic.

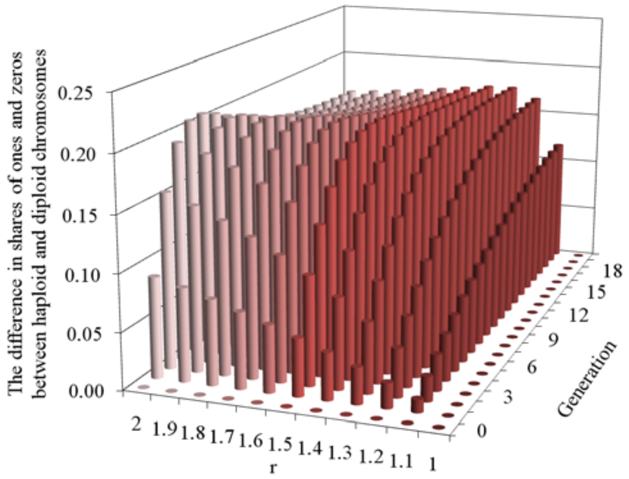

Fig. 1. *The difference in loss rate of inferior genes between haploid and diploid chromosomes for coefficient of mutation 0.0033*

It can be seen from Fig. 1 that during evolution haploid chromosomes lose more weaker genes compared to diploid chromosomes. The process continues until population reaches a steady state after which the weaker genes cannot be lost. The steady state is directly proportional to the coefficient of mutations (in Fig. 1 mutation coefficient is 0.0033). If the coefficient of mutations is greater, the weaker genes are preserved in the population. In the extreme case when the coefficient of mutation is 1 (random search) the proportion of the weaker genes (zeros), with some fluctuations, remains at the level of 50%. Considering the advantages provided by diploid representation of chromosomes and the fact that the Hamilton similarity operates exclusively with individuals with a diploid chromosome the way the proposed algorithm uses diploid chromosomes is more than obvious.

1.2 Hamilton Similarity

Determining the degree of similarity between individual members of the population is a challenging problem, especially in the case of the JSSP. The problem arises from the fact that conversion from genotypic to phenotypic representation is carried out using the so called active schedule. The problem with active schedule approach is that there may be two fundamentally different genotypes with identical or very similar phenotypic representations. On the other hand, two nearly identical genotypes can give completely different phenotypes. For this reason, genotypic and phenotypic comparison of individuals is difficult and unreliable so a new way of comparison should be found. The proposed algorithm uses the so-called Hamilton similarity based on a statistically predictable number of genes that two individuals have in common. The method is named after the British evolutionary biologist Hamilton [7], who in two of his famous papers laid down the mathematical basis for determining the occurrence probability of a particular gene in the population. According to the theory, all genes of one individual are by the same ratio inherited from both parents. Diploid organisms on each position within the chromosome have two complementary genes, of which only one would express itself and affect the phenotype of the organism. Since diploid organisms have two genes at each position in the chromosome, during the creation of new offspring each parent gives only one gene from each position. From the mentioned, it can be concluded that diploid organisms inherit exactly half of their genes from each parent. This regularity is a backbone of the proposed Hamilton similarities because similarity is determined exclusively by origin.

If some rare varieties are disregarded, such as the fact that genes could disappear or appear in the population due to mutation, it can be argued that parents and offspring have 50% genes in common. The next-generation of offspring has 25% of genes in common with the first generation and so on. This regularity can be formulated as:

$$genes\,in\,common = \left(\frac{1}{2}\right)^n \quad , \tag{2}$$

where n is generational distance.

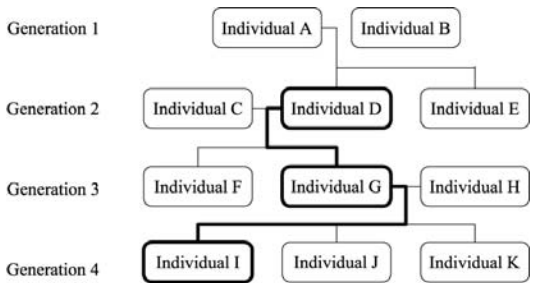

Fig. 2. *Generation distance between direct ancestors*

The example in Fig. 2 clearly shows that the generational distance between individual D and individual I are two generations. According to Eq. (2) those two individuals have 25% of their genes in common.

Determining the generational distance imposes another question. What if two individuals are not direct ancestors? In such case, a more general form of Hamilton's theory must be applied. Specifically, the generational distance between two individuals is determined by adding together distances between each offspring from their nearest common ancestor. Two individuals who are related must share at least one common ancestor. It is important to note the word "at least", because the existence of two ancestors in common in the same generation has an important implication on the number of genes in common.

A generalized version of the Eq. (2) is as follows:

$$genes\,in\,common = m \cdot \left(\frac{1}{2}\right)^{n} , \tag{3}$$

where m is the number of ancestors in common and it can take values 1 or 2, while n is the sum of the generational distances between each individual and the first ancestor in common. If, for instance, the number of genes in common between individual F and individual K were be found, then the first nearest common ancestor would be detected.

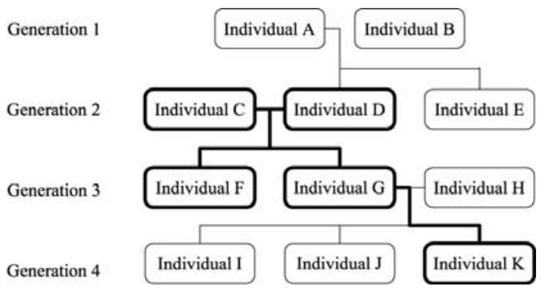

Fig. 3. *Generational distance between not direct ancestors*

From Fig. 3 it can be seen that their nearest ancestor is in generation 2. It is important to note that in generation 2 there are two common ancestors (individual C and individualy D), so coefficient m from the Eq. (3) equals 2. Furthermore, the generational distance between individual F and K and their common ancestors can be obtained. Individual F is one generation away and individual K is two generations away from their common ancestors, so they have 25% of the genes in common:

$$genes\,in\,common = 2 \cdot \left(\frac{1}{2}\right)^{1+2} = 0.25 . \tag{4}$$

1.3 Hamilton Similarity and Fitness Sharing

Sharing [8] is by far the most commonly and most successfully used niching method [9]. According to the theory all individuals located within a niche are forced to share resources that niche provides. Fitness sharing as resource that is shared uses fitness function value. It simply reduces potential value of the fitness function proportional to the number of similar individuals in the population or the number of individuals in a particular niche. For proper operation of the algorithm it is necessary to determine the sharing function between the observed individual and all other individuals in the population. The sharing function is proportional to the similarity of two individuals $d(i,j)$. The similarity of individuals ranges from 0 to 1. If two individuals are identical, then their $d(i,j)$ is 1. If individuals are not identical and their difference exceeds the tolerance, then the value of $d(i,j)$ is 0, which practically means that they do not affect each other because they do not belong to the same niche. All other individuals have similarities between these two extremes. Corrected value f' of the fitness function can be defined as:

$$f' = \frac{f(i)}{\sum_{j=1}^{n} sh(d(i,j))} , \tag{5}$$

where sh is the sharing function that can be defined as:

$$sh = \begin{cases} 1 - \left(\dfrac{d}{\sigma_{share}}\right)^{\alpha} & d < \sigma_{share} \\ 0 & others \end{cases} , \tag{6}$$

where σ_{share} is the threshold and α is the coefficient for fine tuning of individual influence.

In the Hamilton similarity the sharing function does not depend on “hard to define” similarities between individuals but upon generational distance between them, so sharing functions are defined as follows:

$$sh = \begin{cases} \left(\left(\dfrac{1}{2}\right)^{g}\right)^{\alpha} & g < G_{similarity} \\ 0 & others \end{cases} . \tag{7}$$

$G_{similarity}$ is the number of previous generations in which similarity between two individuals is checked. If in the specified number of generations a common ancestor cannot be found, then it is considered that those individuals are not related and it is 0. The Hamilton similarity and the sharing functions computation is a central point of this paper.

The sharing functions depending on generation distance and coefficient α from two individuals from the same generation are displayed in Fig. 4. Changing the value of the coefficient α, mutual influence of individuals within a niche can be fine tuned.

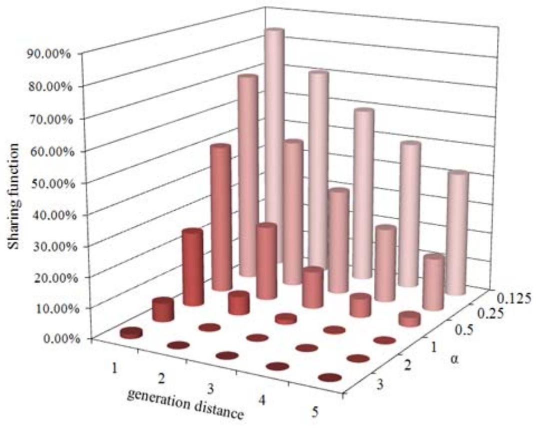

Fig. 4. *Sharing function*

1.4 Domination

In every algorithm that uses diploid chromosomes a problem of domination appears. In order to make the transition from genotypic to phenotypic view possible, for each position within the chromosome, a dominant gene must be determined. This dominant gene will be manifested in the form of phenotypic characteristics. In the literature several domination schemes can be found, but unfortunately they are mostly developed for the binary version of a chromosome. An additional problem with diploid chromosomes is the fact that only indirect representation can be used in order to avoid getting illegal solutions. For this reason a large number of scheduling problems that use direct operation representation cannot be used together with diploid chromosomes. The proposed algorithm uses the so-called "coding with reference list" that always guarantees a correct solution but at the same time allows a very large number of different genes to be at the same position, which implies a rather complicated dominance map. For the purposes of the proposed algorithm dominance map is composed as shown in Table 1.

A dominance map is shown for the chromosome position where only five complementary genes can exist. The proposed dominance map guarantees that every possible gene has an equal number of dominant and recessive combinations.

Table 1. *Dominance map*

	1	**2**	**3**	**4**	**5**
1	1	1	3	1	4
2	1	2	2	4	2
3	3	2	3	3	5
4	1	4	3	4	4
5	5	2	5	4	5

2 SELECTION METHODS AND GENETIC OPERATORS

2.1 Selection Methods

Selection is a process that determines how many times a particular individual will become a parent which directly affects the number of offspring to which it will transmit its genes. The process can be divided into two phases:

- Determining the number of attempts certain individual can expect to become parent.
- Converting the potential number of attempts in the actual number of offspring.

Determining the number of attempts to become a parent is actually a process of converting an individual's fitness function into expected number of participation in creating the next generation. Since the expected number of participation is almost always a decimal number, the second part of selection process relates to a conversion of decimal number into integer, which refers to the actual number of offspring.

Since several different selection processes are proposed by different authors some kinds of criteria have to be established in order to compare them. The best known criteria were defined by Baker in 1987 [10]. Baker has defined three parameters:

- Bias - the absolute difference between actual and expected probability of selecting a particular individual. Optimal value is zero. That means that there is no difference between the expected number of participation in the reproduction process (the first step of selection) and the number of offspring (the second step of selection).
- Spread - the range between theoretically minimal and theoretically maximal number of possible participation in the creation of the next generation individual can accomplish. The actual number of attempts

must be within the defined range. Narrow range is desirable.
- Complexity - some selection methods are extremely complex and require large CPU time. The method that does not contribute to the complexity of the genetic algorithm is considered to be the better.

For the purposes of the proposed algorithm three selection methods are considered: Stochastic Sampling with Replacement, Tournament selection and Stochastic Universal Sampling. Due to minimal bias and zero spread Stochastic Universal Sampling has been chosen as the selection method.

2.2 Genetic Operators

Mutation operators and crossover operators belong to the category of genetic operators. Crossover operators can be considered most important as they are responsible for the exchange of genetic material. Classical crossing operators such as:
- one-point crossover,
- two-point crossover,
- *n*-point crossover and
- uniform crossover

are applicable only on chromosomes that operations in schedule represent in indirect way. When direct representation is used, in many cases illegal solutions are obtained. Since the proposed algorithm uses coding with reference list (indirect representation) the use of classical crossover operators is allowed. For the purposes of the proposed algorithm one and two-point crossover operators are chosen.

The main purpose of the mutation operator is an attempt to preserve genetic diversity of the population. If the population is small, it is likely that dominance of the best adopted individual will occur very quickly and the genetic material of the population would be decimated. The problem can be solved in two ways; by increasing the frequency of mutation, which indirectly leads to more stochastic algorithm or by increasing the size of the population. Previous studies have shown that mutation should remain relatively small and time-regulated. In the proposed algorithm the frequency of mutation is kept below 0.33%.

3 FITNESS SCALING

Fitness scaling has an extremely important role for niching. Indeed it can be said that for most fitness functions, if fitness sharing is used, employment of fitness scaling is almost necessary. It is not realistic to expect niching if there is no significant difference between the fitness function values of particular individuals [8]. If the fitness function is configured in such a way that the fitness functions of all individuals are around the same value, the selection process would become fully stochastic. The problem can be solved in numerous ways but two of them are particularly interesting:
- linear scaling,
- exponential scaling.

Both methods are applied in the proposed algorithm using the time dependent scaling component. If exponential scaling is used without the time dependent component it may happen that in the early phase of evolution, dominance of the currently best individual occurs and the algorithm can end up in one of the local optimums. Using the time dependent exponential scaling early dominance of certain individuals is prevented because exponential scaling progressively increases in the later stages of evolution when the niches are already well defined. Unfortunately, in the last phase of the evolution, when the generation is comprised of mostly well-adjusted individuals, the selection process can become stochastic again. In order to solve this problem the exponent value is directly dependent on the value of the fitness function. This means that individuals with larger values of fitness function are further enhanced with a higher exponent values, so a better adopted individual in the final stages of the evolution is more often selected for reproduction. In this way all niches are dismantled in order to enable the algorithm to exploit the best niche. The proposed algorithm uses the following Fitness Scaling function:

$$f' = (f-\delta)^{\left(\frac{G}{G_{tr}}\right)\left(\frac{f-f_{\min}}{f_{\max}-f_{\min}}\right)+\beta}, \qquad (8)$$

where:

δ	coefficient of linear scaling. It can take values between 0 and the minimum of fitness function,
G_{tr}	transition generation. The generation after which higher selection pressure is applied,

G	current generation,
f_{min}	minimum value of fitness function in the current generation,
f_{max}	maximum value of fitness function in the current generation,
f	fitness function value,
f'	corrected fitness function value,
β	coefficient of basic exponential scaling.

4 EXPERIMENT

The experiment described in this section is conducted on a well known mt10 benchmark problem. The problem is particularly significant as it is used in almost any research connected with the JSSP. JSSP is a problem of scheduling set of jobs on a set of machines. JSSP cannot be easily solved, because, from the combinatorial point of view, it is classified as NP-hard, which means that it cannot be exactly solved even for a relatively small number of operations and machines.

The purpose of the experiment was to show that the genetic algorithm can function very well without context sensitive information, and that results are quite reliable and close to global optimum.

The experiment was conducted under the following conditions:

- the number of generations: 300,
- population size: 300,
- frequency of mutation: 0.0033,
- frequency of crossover: 1,
- $G_{similarity}$: 3 generations,
- δ linear scaling: 0,
- G_{tr} transition generation: 150,
- β basic exponential scaling: 6,
- α power of sharing function: 1.

Table 3. *Statistical data*

Number of evolutions	Hamilton	SGA
	600	600
Standard deviation	12.9812	22.26397
Main	974.4033	1027.142
Median	976	1027
Max	1012	1086
Min	937	962

In order to make the experiment statistically reliable 600 evolutions were conducted and compared to the results from simple genetic algorithm (SGA). The number of occurrences of a particular solution is shown in Fig. 5.

The experiment has clearly showed that proposed algorithms demonstrated superior results compared to the SGA.

4.1 Influence of Evolutionary Parameters

For the purposes of the experiment coefficients of mutation and crossover are kept constant. These coefficients are considered irrelevant for the outcome of the experiment. On the other hand, G and β proved themselves extremely important. Initial tests have shown that if G_{tr} is too small, the selection pressure becomes too high too early and the process of forming niches is seriously compromised. If G_{tr} is large, the algorithm does not have enough time to exploit the best niches in the final stage of the evolution.

The sole purpose of coefficient β is to increase selection pressure which indirectly leads to the rapid creation of niches. There is no specific rule for determining the value of the coefficient β, but the rule of thumb is that a larger population can have higher values of β.

Table 2. *Muth and Thompson's 10 x 10 problem (mt10)*

Job	Machine (processing time)									
1	1 (29)	2 (78)	3 (9)	4 (36)	5 (49)	6 (11)	7 (62)	8 (56)	9 (44)	10 (21)
2	1 (43)	3 (90)	5 (75)	10 (11)	4 (69)	2 (28)	7 (46)	6 (46)	8 (72)	9 (30)
3	2 (91)	1 (85)	4 (39)	3 (74)	9 (90)	6 (10)	8 (12)	7 (89)	10 (45)	5 (33)
4	2 (81)	3 (95)	1 (71)	5 (99)	7 (9)	9 (52)	8 (85)	4 (98)	10 (22)	6 (43)
5	3 (14)	1 (6)	2 (22)	6 (61)	4 (26)	5 (69)	9 (21)	8 (49)	10 (72)	7 (53)
6	3 (84)	2 (2)	6 (52)	4 (95)	9 (48)	10 (72)	1 (47)	7 (65)	5 (6)	8 (25)
7	2 (46)	1 (37)	4 (61)	3 (13)	7 (32)	6 (21)	10 (32)	9 (89)	8 (30)	5 (55)
8	3 (31)	1 (86)	2 (46)	6 (74)	5 (32)	7 (88)	9 (19)	10 (48)	8 (36)	4 (79)
9	1 (76)	2 (69)	4 (76)	6 (51)	3 (85)	10 (11)	7 (40)	8 (89)	5 (26)	9 (74)
10	2 (85)	1 (13)	3 (61)	7 (7)	9 (64)	10 (76)	6 (47)	4 (52)	5 (90)	8 (45)

Algorithms with high β and small population size tend to get stuck in one of the local optimums.

The sole purpose of coefficient β is to increase selection pressure which indirectly leads to a rapid creation of niches. There is no specific rule for determining the value of the coefficient β, but the rule of thumb is that larger population can have higher values of β. Algorithms with high β and small population size tend to get stuck in one of the local optimums.

In addition to the coefficient β, coefficient δ is also considered. Since the minimum of fitness function should be known in advance in order to correctly estimate the value of δ, it was rendered completely useless. Coefficient δ was set to 0 for all tests.

Finally, coefficient α was used for controlling the niche capacity. For small values of α, the capacity of niche is significantly reduced. Since a relatively small population is used, the capacity of the niche should also be small in order to explore larger partition of the search space.

4.2 Additional Experiments

The proposed algorithm was also tested on all 40 "la" benchmark problems [11]. The evolutionary parameters used for each group of problems are shown in Table 4. Each test was run 50 times and statistical results are given in Table 5. Due to a relatively small population size, and therefore small genetic diversity, the algorithm had difficulties finding optimal solution for large problems. A small population was chosen in order to speed up the algorithm. Even for a relatively small population, the algorithm has found 18 out of 37 known global optimums.

5 CONCLUSION AND FURTHER RESEARCH

Two conclusions can be drawn from the conducted experiments. First, the use of the Hamilton similarity allows the application of the algorithm on various problems with minor modification. This means that context sensitive information is no longer necessary in order to determine the similarity between two individuals. Second, time dependent scaling enables an increase in the selection pressure only when niches are well defined which implicitly helps algorithm to avoid one of the less desirable local optimums, which is clearly shown by the results of an experiment.

Although the algorithm showed very positive results there is still plenty of room for improvement. First of all, the dominance map is defined quite arbitrarily and it is not subjected to the process of evolution, which is a problem worth testing [12]. Furthermore, the steady-state version of the algorithm should also be tested. In nature it is common for parents to defend their offspring or to delegate part of their resources to them. Since the proposed algorithm in each generation is exchanged for the whole population, the impact of the surviving parents on the algorithm performance has not been tested.

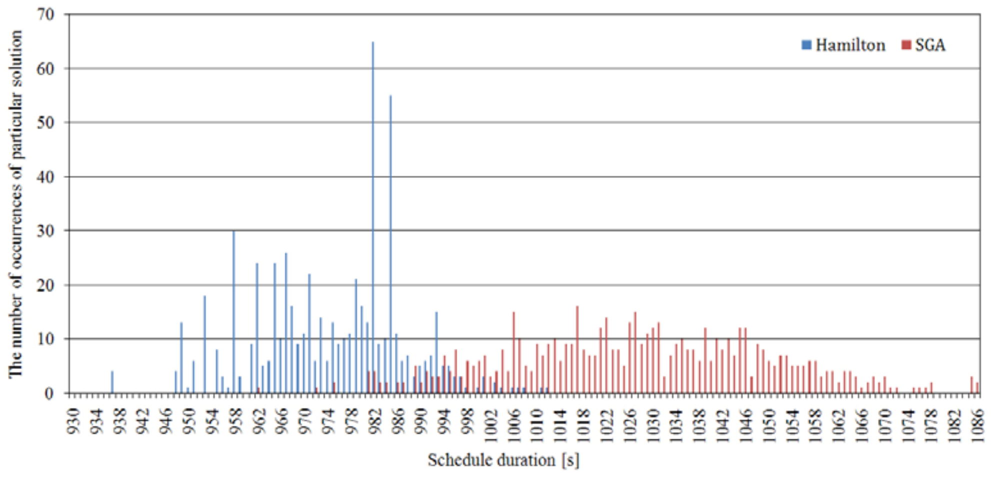

Fig. 5. *The histogram of the best makespans obtained after 600 trials for the mt10 problem*

Table 4. *Evolutionary parameters (la benchmark problems)*

	G	G_{tr}	β	α	$G_{similarity}$	Mutation	Crossover	Pop. size
la01-la05 (10x5)	300	150	4	1	5	0.0033	0.95	300
la06-la10 (15x5)	400	200	4	1	5	0.0033	0.95	300
la11-la15 (20x5)	400	200	4	1	5	0.0033	0.95	300
la16-la20 (10x10)	400	200	4	1	4	0.0033	0.95	300
la21-la25 (15x10)	500	250	4	1	5	0.0033	0.95	300
la26-la30 (20x10)	750	375	4	0.2	5	0.0033	0.95	300
la31-la35 (30x10)	1000	600	4	0.2	5	0.0033	0.95	300
la36-la40 (15x15)	800	480	4	0.2	5	0.0033	0.95	300

Table 5. *Statistical results (la benchmark problems)*

	la01	la02	la03	la04	la05	la06	la07	la08	la09	la10
Main	666	669.4	617.7	606.7	593	926	890	863	951	958
Median	666	669.5	617	607	593	926	890	863	951	958
Best	**666**	657	609	593	**593**	**926**	**890**	**863**	**951**	**958**
Worst	666	682	622	612	593	926	890	863	951	958
Optimum	666	655	597	590	593	926	890	863	951	958
σ	0	6.217	2.184	4.834	0	0	0	0	0	0
	la11	la12	la13	la14	la15	la16	la17	la18	la19	la20
Main	1222	1039	1150	1292	1207	977.9	787.5	875.7	875	922.2
Median	1222	1039	1150	1292	1207	979	787	875	873	918
Best	**1222**	**1039**	**1150**	**1292**	**1207**	956	785	855	863	913
Worst	1222	1039	1150	1292	1207	982	804	884	886	944
Optimum	1222	1039	1150	1292	1207	945	784	848	842	902
σ	0	0	0	0	0	6.129	3.435	6.506	4.793	9.324
	la21	la22	la23	la24	la25	la26	la27	la28	la29	la30
Main	1093.8	980.8	1041.3	998.7	1026.2	1257.2	1300.1	1279.2	1248.8	1396.4
Median	1096	982.5	1039	1002	1024	1265.5	1299.5	1278	1251	1397.5
Best	1068	956	**1032**	977	1008	1237	1287	1252	1212	1367
Worst	1125	994	1061	1020	1064	1278	1315	1307	1276	1418
Optimum	-	927	1032	935	977	1218	-	1216	-	1355
σ	14.42	7.913	8.484	8.875	11.230	12.389	8.343	14.365	13.757	13.272
	la31	la32	la33	la34	la35	la36	la37	la38	la39	la40
Main	1785.5	1852.3	1720.3	1733.3	1898.1	1319.9	1462.1	1277.3	1309.9	1286.0
Median	1784	1850	1719	1730	1898	1322	1460	1278	1312	1284
Best	**1784**	**1850**	**1719**	**1721**	**1888**	1296	1449	1252	1268	1270
Worst	1791	1964	1736	1761	1928	1335	1490	1303	1335	1326
Optimum	1784	1850	1719	1721	1888	1268	1397	-	1233	1222
σ	2.158	3.815	4.433	10.632	9.649	10.099	10.873	12.967	14.753	11.357

6 REFERENCES

[1] Goldberg, D. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison Wealey Longman, Inc, p. 412.

[2] Knjazew, D. (2002). *OmeGA: a competent genetic algorithm for solving permutation and scheduling problems*. Kluwer Academic Publisher, p. 152.

[3] Grant, D. (2005). *A Comparison of localized and global niching methods. SIRC 2005 - The 17th annual colloquium of the Spatial Information Research Centre.* University of Otago, Dunedin, New Zealand.

[4] Kowalczuk, Z., Białaszewski, T. (2006). *Niching mechanisms in evolutionary computations*. Faculty of electronics, telecommunications and computer science, Gdansk University of Technology.

[5] Goldberg, D., Smith, R. (1987). Nonstationary function optimization using genetic algorithms with dominance and diploidy. *International Conference on Genetic Algorithms,* Cambridge, MA, p. 59-68.

[6] Mohfoud, S.W. (1995). *Niching Methods for Genetic Algorithms*. PhD. Dissertation, University of Illinois at Urbana-Champaign, Urbana.

[7] Dowkins, R. (2006). *The Selfish Gene*. Oxford University Press, p. 384.

[8] Ursam, R.K. (2001). *When Sharing Fails*, EvALife project group, Department of Computer Science, University of Aarhus.

[9] Goldberg, D.E., Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. *Proceedings of the Second International Conference on Genetic Algorithms and their application*, p. 41-49.

[10] Baker, J.E. (1987). Reducing bias and inefficiency in selection algorithms. Genetic algorithms and their applications. *Proceedings of the second International Conference on Genetic Algorithms.*

[11] Lawrence, S. (1984). *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (supplement).* Technical report, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh.

[12] Yang, S. (2007). Learning the dominance in diploid genetic algorithms for changing optimization problems. *Proceedings of the 2nd Int. Symp. on Intelligence Computation and Applications,* China University of GeoSciences Press, p. 157-162.