Fine Grained Tournament Selection for the Simple Plant Location Problem

Vladimir Filipović[1], Jozef Kratica[2], Dušan Tošić[1], and Ivana Ljubić[3]

[1] University of Belgrade
Faculty of Mathematics
Studentski trg 16,. 11000 Belgrade
YUGOSLAVIA
{dtosic | vladaf}@matf.bg.ac.yu

[2] Serbian Academy of Sciences and Arts
Mathematical Institute
Kneza Mihaila 35/I, 11001 Belgrade, p.p. 367
YUGOSLAVIA
jkratica@mi.sanu.ac.yu
URL: http://www.geocities.com/jkratica/

[3] Vienna University of Technology
Institute for Computer Graphics
Favoritenstrasse 9 11/186, A-1040 Vienna
AUSTRIA
ljubic@apm.tuwien.ac.at
URL: http://www.apm.tuwien.ac.at/people/Ljubic.html

ABSTRACT

The simple plant location problem is considered and a genetic algorithm is proposed to solve this problem. Genetic algorithm that solves simple plant location problem uses an improvement of tournament selection, called fine grained tournament selection, as selection operator. New operator is generalization of classical tournament selection, that keeps all good features of classical tournament selection. By using the developed algorithm it is possible to solve SPLP with more than 1000 facility sites and customers. Computational results are presented and compared to rank-based and classical tournament selection.

1. INTRODUCTION

Combinatorial optimization problems are important part of the global optimization. One such problem is simple plant location problem (SPLP). This problem is also known as uncapacitated warehouse location problem or uncapacitated facility location problem.

1.1. Problem Definition

Consider a set $I = \{1,...., m\}$ of candidate sites for facility location, and a set $J = \{1,...., n\}$ of customers. Each facility $i \in I$ has a fixed cost f_i. Every customer $j \in J$ has a demand b_j, and c_{ij} is the unit transportation cost from facility i to customer j. Without a loss of generality we can normalize the customer demands to $b_j = 1$ [5].

It has to be decided:
- facilities to be established and
- quantities to be supplied from facility i to customer j

such that the total cost (including fixed and variable costs) is minimized.

Mathematically, the SPLP is formulated on the following way:

$$\min \left(\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{m} f_i y_i \right) \qquad (1)$$

subject to:

$$\sum_{i=1}^{m} x_{ij} = 1, \text{for every } j \in J; \qquad (2)$$

$$0 \le x_{ij} \le y_i \text{ and } y_i \in \{0,1\}, \text{ for every } i \in I \text{ and every } j \in J; \qquad (3)$$

where:

x_{ij} represents the quantity supplied from facility i to customer j,

y_i indicates whether facility i is established ($y_i = 1$) or not ($y_i = 0$).

Let the set of established facilities be $E = \{ i \mid y_i = 1 \}$ with cardinality $e = |E|$.

Although some special cases of the SPLP are solvable in polynomial time [16, 33, 27] in general, it is a NP-hard problem [12]. Genetic algorithms are successfully used to solve some NP-hard problems in [23, 25].

1.2. Literature Survey

Many approaches have been proposed for solving SPLP. The presentation of all important contributions relevant to SPLP lies beyond the scope of this paper. Some important survey articles are [7, 9, 12]. We are going to mention only several efficient and well-known methods to solve SPLP.

The DUALOC algorithm by Erlenkotter [8] has been the fastest algorithm for solving SPLP for a long time. This algorithm is based on a linear programming dual formation (LP dual) in condensed form that evolved in simple ascent and adjustment procedure. If ascent and adjustment

procedures do not find optimal solution, Branch-and-Bound (BnB) procedure completes the solution process.

Guignard [10] proposed to strengthen the separable Lagrangean relaxation of the SPLP by using Benders inequalities generated during a Lagrangean dual ascent procedure. The coupling that technique with a good primal heuristic could reduce integrality gap.

In [14] Simao and Thizy presented streamlined dual simplex algorithm designed on the basis of a covering formulation of the SPLP. Their computational experience with standard data sets indicates the superiority of dual approaches.

Körkel [11] showed how to modify a primal-dual version of Erlenkotter's exact algorithm to get an improved procedure. The computational experience with large-scale problem instances indicated that speedup to DUALOC is significant (more than one order of magnitude).

Coon and Cornuejols present a method based upon the exact solution of the condensed dual of LP relaxation via orthogonal projections in [6].

In [37] Holmberg used a primal-dual solution approach based on decomposition principles. He fixed some variables in the primal subproblem and relaxed some constraints in the dual subproblem. By fixing their Lagrange multipliers, both of these problems become easier to solve than the original one. The computational tests showed advantageous in comparison to the dual ascent method of Erlenkotter.

2. GENETIC ALGORITHMS (GA)

Under the most frequent classification, genetic algorithms and some related techniques, together with Fuzzy logic and Neural networks, are part of so called Soft computing.

GA are based on Darwin's evolution theory and Mendel's laws of inheritance. If Darwin's theory describes natural processes, then GA imitate natural processes attempting to solve particular problem. Menu important consequences are coming from this claim:
- biological theories are bases of GA;
- imitating of natural mechanisms is, in fact, imitating of biologist's view on that natural mechanisms, and any change in relevant biologist theories should be reflected on GA;
- modifications of GA should be inspired not only with trying to achieve better results, but primarily with trying to better imitate occurrences and processes in nature.

In other words, among authors in GA area is widely accepted [1] that only those modifications, which have analogy in nature, can significantly improve theirs performances.

According to Darwin, individuals in population are competing for resources. Facts that lie in essence of natural selection process are:
- better fitted individuals more often survive and they have stronger influence on forming new generations;
- individual in new generation is formed by recombination of parent's genetic material;
- from time to time mutation (random change of genetic material) takes place.

Great deals of popularity of GA lies in fact that getting new solutions has, indeed, parallel nature. Holland [2] first observed that parallel nature of reproductive paradigm and inherent efficiency of parallel processing.

Also, GA are easy to hybridize: they easy can be combined with other algorithms that solve specific problem. New, hybrid, algorithm has good features of both base algorithms.

2.1. Definition

GA has following structure:
- **search space** – space of all possible solutions;
- **population** – set of actual candidates for solution; elements of population are called items, or search nodes, or points in search space;
- **string space** – space which contains string representations of items in population;
- **functions for conversion** between points in search space and points in string space (coding and decoding);
- **set of genetic operators** for generating new strings (and new items);
- **fitness function** – it evaluate fitness (degree of adaptation) for each item in population;
- **stochastic control** of genetic operators.

Basic steps in GA are:
1. **Initialization** – generating initial population by random sampling.
2. **Evaluation** – calculating fitness for all items in population.
3. **Selection** – choosing surviving items in population, according to values of fitness function.
4. **Recombination** (includes crossover and mutation) – changing item's representation.
5. **Repeating steps 1.-4**. until fulfilling finishing criteria.

Features that made distinction between GA and other problem-solving methods are:
- algorithm work with codes that represents parameters, not with parameters themselves;
- algorithm search in several points (population), therefore it is very robust;
- algorithm use probabilistic transition rules, rather than deterministic;
- algorithm (in its pure form) doesn't exploit additional information about problem's nature.

Simple GA (SGA) is start point for all other modifications of GA. SGA has following operators: one-point crossover, one-point mutation and roulette selection. Individuals in SGA are represented with binary strings.

One point crossover is exchange of genetic material between two individuals from particular (randomly chosen) position. This position is called crossover point.

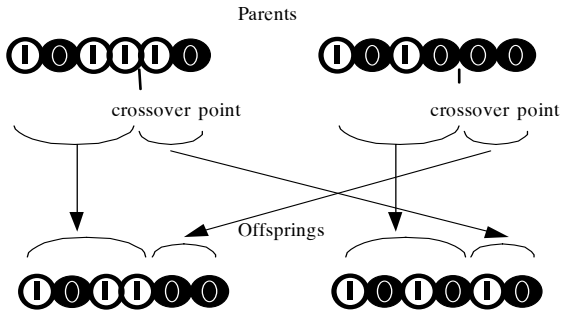

Fig 1. One-point selection operator

One point mutation is simple complementation of one (randomly chosen) bit in genetic material of the selected individual.

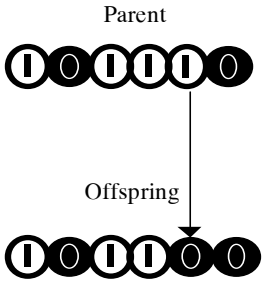

Fig 2. One-point mutation

Mutation is, in classical interpretation [2], less significant operator than the crossover. Probability of mutation is usually for two or three orders of magnitude smaller than probability of crossover. It should be noted that nowadays many authors denied traditional estimate that crossover is much more significant than mutation [18, 28, 29].

Another operator that is very important in GA is selection. Selection operator doesn't modify individuals in population. It selects individuals that will survive and pass their genes to next generation. SGA uses roulette (also called proportional) selection as selection operator. In roulette selection, individual's probability to survive is equal to fitness of individual divided with average fitness of population.

2.2. Literature Survey

Some important survey books and articles about GA are [1, 3, 4, 39, 36].

Different modifications of pure GA are used to improve performances:
- during forming of string that represents individual [1];
- during coding (transforming search space into string space) – Gray coding, multidimensional coding [1, 36];
- during computing of a fitness value for each individual and during executing one iteration in GA [24, 35];
- during designing of crossover [31, 21], mutation [1] and selection operators [18, 17, 32, 26];
- during replacement of generation – steady state, elitist strategy [1];
- during setting stochastic parameters that controls applying of GA operators [22, 30, 18, 15];
- creation of a hybrid between GA and some other heuristic [20, 38].

3. TOURNAMENT AND FINE GRAINED TOURNAMENT SELECTION

3.1. Tournament Selection

Tournament selection is one of the most popular selection operators. Its popularity grows in recent time, because this operator is excellent suited for applying in parallel genetic algorithms.

In tournament selection, element of population is chosen for passing into next generation if it is better (has better fitness value) than several randomly selected opponents. Tournament size N_{tour} is selection parameter.

Algorithm looks like:
Input: Population a (size of a is n), tournament size N_{tour}, $N_{tour} \in N$
Output: Population after selection a' (size of a' is n)
Tournament_Selection:
begin
    for i := 1 to n do
        a_i' := best fitted item among N_{tour} elements randomly selected from population;
    return a';
end;

Running time for this algorithm is $O(n*N_{tour})$. Usually is $N_{tour} << n$, so algorithm is linear, time is O(n). Thus, tournament selection differs from explicitly ranking schemes such are linear ranking and exponential ranking, because it doesn't need firstly to sort population during its work.

Tournament selection allows parallel execution during choosing members of new generation. It is, just like ranking schemes, invariant to translation (adding same value to the fitness of all item in population) and scaling (multiplying fitness of all items in population with some value).

But, tournament selection doesn't allow precise ratio setting between exploration and exploitation [19], what is crucial for using this selection method in practice. Ratio between exploration and exploitation governs search process in GA. Level of exploration (looking for new

solutions) and exploitation (using previously acquired knowledge) is in tournament selection determined with N_{tour}, that can be one of few integer values (usually 2-3 values are good candidates). Often happens that with smaller tournament size search process runs too slow, and that with bigger tournament size it runs too fast and premature converges.

More details about theoretical features of tournament selection can be found in [34].

3.2. Fine Grained Tournament Selection

In [34] is described one improvement of the tournament selection, named fine grained tournament selection

Instead of integer parameter N_{tour} (which represents tournament size), new operator allows real valued parameter F_{tour} – wanted average tournament size. This parameter governs selection procedure, so average tournament size in population should be as close as possible to it.

Just like in tournament selection, item is choused if it is better than its randomly choused opponents. But, this time tournament size isn't fixed within population. So, during one selection step will be held tournaments with different number of competitors.

Current implementation of fine grained tournament selection during one step of execution holds tournaments with difference among sizes less or equal to 1. Sizes of n held tournaments are chosen with wish that its average becomes as close as possible to real value F_{tour}.

Algorithm looks like:
Input: Population a (size of a is n), wished average tournament size F_{tour}, $F_{tour} \in R$
Output: Population after selection a' (size of a' is n)
Fine_Grained_Tournament_Selection:
begin
$F_{tour}^{-} := trunc(F_{tour})$
$F_{tour}^{+} := trunc(F_{tour}) + 1$
$n^{-} := trunc (n * (1 - (t - trunc (F_{tour}))))$
$n^{+} := n - trunc (n * (1 - (t - trunc (F_{tour}))))$
/* tournaments with size F_{tour}^{-} */
for i := 1 to n^{-} do
a_i' := best fitted item among F_{tour}^{-} elements randomly selected from population;
/* tournaments with size F_{tour}^{+} */
for i := n^{-}+1 to n do
a_i' := best fitted item among F_{tour}^{+} elements randomly selected from population;
return a'
end;

Fine grained tournament selection provides that ratio between exploration and exploitation can be set very precise. This selection method (like tournament selection) also has analogy with some processes that are happening in nature.

Theoretical properties of new selection operator are also discussed in [34]. It is shown that new operator preserves good features of classical tournament selection (algorithm has linear time of execution, it allows execution in parallel, it is invariant to translation and scaling, etc.).

In that work is proved that classical tournament selection is special case of fine grained tournament selection.

4. GA IMPLEMENTATION

Implementation used for comparison among several selection methods is based on binary representation, uniform crossover, simple mutation, steady-state generation replacement with elitist strategy and caching of GA.

Three types of selection methods are compared:
– Rank based selection, with rank 2.5 for the best individual down to 0.7 for the worst, by step 0.012. For SPLP, this selection scheme successfully prevents premature convergence in local optima and loosing the genetic material.
– Classical tournament selection with tournament size $N_{tour} \in \{5, 6\}$.
– Fine grained tournament selection with wanted average tournament size $F_{tour} \in \{4.5, 5.5, 5.6, 5.8, 6.2, 6.4\}$

Uniform crossover is performed, that provides minimal disruption of individual's genes. It is important part for success of GA, because interaction among genes in individual's gene code isn't too big.

Simple mutation is used and its run-time is improved by realization through normal distribution. Mutation rate depends on problem size and it exponentially decreases from 0.4/n to 0.15/n.

Steady-state generation replacement with elitist strategy is used. In each generation, only 1/3 of population is replaced. Remaining 2/3 of population is directly passed to the next generation. Those elitist individuals don't need reevaluation (their objective value is already evaluated). The maximal number of generation is 2000*n. If best individual is same for 1000*n generations, GA terminates.

Population size is 150 individuals and population is randomly initialized in first generation. Binary code is used for encoding of variables.

Finally, run-time performance of the GA is improved by caching technique [13]. Least recently used (LRU) strategy, which is simple but effective, is used for caching GA. Caching is implemented by hash-queue data structure.

5.COMPUTATIONAL RESULTS

Appliance of the fine grained tournament selection on SPLP gives better results in comparison to both rank-based and classical tournament selection (see Tables 1-4).

Results from these tables are obtained running GA on Pentium III /600MHz PC, with 330 megabytes of psychical memory.

All these values are averages determined from 20 independent runs per problem instance.

Due to better visibility, results are summary displayed per instance group. Columns in tables represent problem instance groups and rows represent selection type.

Tables contain results for: rang based selection, tournament selection (N_{tour} = 5, 6) and fine grained tournament selection (F_{tour} = 4.5, 5.5, 5.6, 5.8, 6.2, 6.4).

Every cell in tables has two values. Upper value is average number of generations and lower value is average running time (in seconds).

In all executions of every program instance is obtained same result that is equal to optimal or previously best known solution. Best times for each instance group in table are shaded.

Table 1. Comparison of selection operators for ORLIB instances 41-74 and 81-104

Selection	41-74	81-104
r. b.	17.7	33.6
	0.10	0.17
f. g. t. (F_{tour}=4.5)	10.4	34.4
	0.07	0.17
t. (N_{tour}=5)	9.1	49.1
	0.06	0.21
f. g. t. (F_{tour}=5.5)	9	53.5
	0.06	0.24
f. g. t. (F_{tour}=5.6)	8.8	41.3
	0.05	0.18
f. g. t. (F_{tour}=5.8)	8.9	46.6
	0.05	0.20
t. (N_{tour}=6)	8.8	67.1
	0.05	0.30
f. g. t. (F_{tour}=6.2)	8.5	77.1
	0.05	0.34
f. g. t. (F_{tour}=6.4)	8.8	87.1
	0.05	0.38

Table 2. Comparison of selection operators for ORLIB instances 111-134 and A-C

Selection	111-134	A-C
r. b.	109.2	1328
	0.51	16.28
f. g. t. (F_{tour}=4.5)	145.3	1633
	0.65	16.78
t. (N_{tour}=5)	128.9	1433
	0.52	12.72
f. g. t. (F_{tour}=5.5)	136.6	1209
	0.62	12.24
f. g. t. (F_{tour}=5.6)	136.3	1694
	0.55	12.47
f. g. t. (F_{tour}=5.8)	151.2	2347
	0.61	16.81
t. (N_{tour}=6)	146	1078
	0.65	10.74
f. g. t. (F_{tour}=6.2)	164.4	1890
	0.73	18.22
f. g. t. (F_{tour}=6.4)	148.6	2076
	0.66	20.06

It becomes clear from obtained results that almost all ORLIB instances haven't sufficient size to properly test behavior of algorithm on large-scale instances. Thus, selection strategies are tested on problem instances generated and described in [39].

Table 3. Comparison of selection operators for generated instances

Selection	MO	MP	MQ	MR	MS
Rank	112.4	181.7	269.9	423.7	879.4
based	0.59	1.18	2.69	4.68	23.18
Fine G. T.	76.8	131.6	205.9	347	746.9
4.5	0.46	0.83	1.51	3.47	17.24
T. 5	85.2	128.6	206.2	357	732.4
	0.36	0.67	1.28	3.14	15.78
Fine G. T.	96.4	127.9	208.6	340.6	756.1
5.5	0.49	0.79	1.51	3.31	16.77
Fine G. T.	83.6	131.5	192.1	343.4	723.6
5.6	0.35	0.55	0.93	2.52	14.47
Fine G. T.	92.2	122.8	204.4	345.7	766.2
5.8	0.38	0.51	0.99	2.53	14.94
T. 6	96.7	131.3	210.4	332.4	743.8
	0.49	0.81	1.50	3.19	16.31
Fine G. T.	83.5	122.6	207	345.6	740.5
6.2	0.43	0.76	1.47	3.31	16.16
Fine G. T.	87.4	130.7	200.3	340.7	721.3
6.4	0.44	0.81	1.42	3.24	15.65

In most cases, best results (or results that are very close to best ones) are produced by fine grained tournament selection, with F_{tour} = 5.6. Improvement to other selection method is significant (usually 10%-20%) and for some (mainly large-sized) instances it is even bigger.

6. CONCLUSION

In this papers is presented GA implementation for solving simple plant location problem. Such approach is very suc-

cessful in practice and recommended for the large-scale problem instances (more than 1000 facility locations and customers)

GA implementation uses as selection operator newly designed fine grained tournament selection, generalization of classical tournament selection. New selection operator keeps all good features of classical tournament selection.

For SPLP, problem fine grained tournament selection significantly outperforms both rank based and classical tournament selection.

REFERENCES

[1] D.E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, Reading, Mass., USA, Addison-Weseley Publ. Comp., 1989.

[2] J. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975.

[3] D. Beasley, D.R. Bull and R.R. Martin, "An overview of genetic algorithms, part 1, fundamentals", *University Computing,* vol. 15, no. 2, pp. 58-69, 1993.

[4] D. Beasley, D.R. Bull and R.R. Martin, "An overview of genetic algorithms, part 2, research topics", *University Computing,* vol. 15, no. 4, pp. 170-181, 1993.

[5] J.E. Beasley, "Lagrangean heuristic for location problems", *European Journal of Operational Research,* vol. 65, pp. 383-399, 1993.

[6] A.R. Conn and G. Cornuejols, "A projection method for the uncapacitated facility location problem, *Mathematical Programming*, vol. 46, pp. 273-298, 1990.

[7] P.M. Dearing, "Location Problems", *Operations Research Letters*, vol. 4, pp. 95-98, 1985.

[8] D. Erlenkotter, "A dual-based procedure for uncapacitated facility location", *Operations Research*, vol. 26 pp. 992-1009, 1978.

[9] L.L. Gao, E. Robinson and Jr. Powell, "Uncapacitated facility location: general solution procedure and computational experience", *European Journal of Operational Research*, vol. 76, no. 3, pp. 410-427, 1994.

[10] M. Guignard and A Lagrangean, "Dual ascent algorithm for simple plant location problems", *European Journal of Operational Research*, vol. 35, pp. 193-200, 1988.

[11] M. Koerkel, "On the exact solution of large-scale simple plant location problems", *European Journal of Operational Research***,** vol. 39, pp. 157-173, 1989.

[12] J. Krarup and P.M. Pruzan, "The simple plant location problem: survey and synthesis", *European Journal of Operational Research*, vol. 12, pp. 36-81, 1983.

[13] J. Kratica, "Improving performances of the genetic algorithm by caching", *Computers and Artificial Intelligence*, vol. 18, no. 3, pp. 271-283, 1999.

[14] H.P. Simao and J.M. Thizy, "A dual simplex algorithm for the canonical representation of the uncapacitated facility location problem", *Operations Research Letters*, vol. 8, no. 5, pp. 279-286, 1989.

[15] A. Aizawa and B. W. Wah, "Dynamic control of genetic algorithms in a noisy environment", in *Proc. of the Fifth International Conference on Genetic Algorithms*, pg. 48-55, San Mateo, California, 1993.

[16] A.A. Aqeev and V.S. Beresnev, "Polynomially solvable cases of the simple plant location problem", in *Proceeding of the First Integer Programming and Combinatorial Optimization Conference*, pp. 1-6, 1990.

[17] T. Baeck and F. Hoffmeister, "Extended selection mechanisms in genetic algorithms", in *Proc. of the Fourth International Conference on Genetic Algorithms*, pg. 92-99, San Mateo, California, 1991.

[18] T. Baeck, "Self-adaptation in genetic algorithms", in *Proc. of the First European Conference on Artificial Life*, MIT Press, 1992.

[19] T. Blickle and L. Thiele, "A mathematical analysis of tournament selection", in *Proc. of the Sixth International Conference on Genetic Algorithms*, pp. 9-16, , San Mateo, California, 1995.

[20] E. K. Burke, D. G. Elliman and R. F. Weare, "A hybrid genetic algorithm for highly constrained time tabling problems", in *Proc. of the Sixth International Conference on Genetic Alghorithms*, pg. 605-610, San Mateo, California, 1995.

[21] R. A. Caruana, L. J. Eshelman and D. J. Schaffer, "Representation and hidden bias II: eliminating defining length bias in genetic search via shuffle crossover", in *Machine Learning*, pg. 750-755, 1991.

[22] L. Davis, "Bit-climbing, representational bias, and test suite design", in *Proc. of the Fourth International Conference on Genetic Algorithms*, pg. 18-23, San Mateo, California, 1991.

[23] K.E. De Jong and W.M. Spears, "Using genetic algorithms to solve NP-complete problems", in *Proc. of the Third International Conference on Genetic Algorithms*, pp. 124-132, San Mateo, California,1989.

[24] D. E. Goldberg, K. Deb and B. Korb, "Don't worry, be messy", in *Proc. of the Fourth International Conference on Genetic Algorithms*, pg. 24-30, San Mateo, California, 1991.

[25] S. Khuri, T. Back and J. Heitkotter, "An evolutionary approach to combinatorial optimization problems, in *Proc. of CSC'94*, Phoenix, Arizona, 1994.

[26] T. Kuo and S. Hwang, "A genetic algorithm with disruptive selection", in *Proc. of the Fifth International Conference on Genetic Algorithms*, pg. 65-69, San Mateo, California, 1993.

[27] C. Ryu and M. Guignard, "An exact algorithm for the simple plant location problem with an aggregate capacity constraint", in *Proc. of TIMS / ORSA Meeting*, Orlando, FL 92-04-09, 1992.

[28] D. J. Schaffer and L. J. Eshelman, “On crossover as evolutionary viable strategy”, in *Proc. of the Fourth International Conference on Genetic Algorithms*, pg. 61-68, San Mateo, California, 1991.

[29] W. M. Spears, “Crossover or mutation?”, in *Foundation of Genetic Algorithms 2 - FOGA 2*, pg. 221-239, San Mateo, California, 1992.

[30] W. M. Spears, “Adapting crossover in evolutionary algorithms”, in *Evolutionary Programming IV*, pg. 367-384, 1992.

[31] G. Syswerda, “Uniform crossover in genetic algorithms”, in *Proc. of the Third International Conference on Genetic Algorithms*, pg. 2-9, edited by Schaffer David J, San Mateo, California, 1989.

[32] D. Whitley, “The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best”, in *Proc. of the Third International Conference on Genetic Algorithms*, pg. 116-123, San Mateo, California, 1989

[33] C. De Simone and C. Mannino, “Easy instances of the plant location problem”, Technical Report R-427, Gennaio, University of Roma, 1996.

[34] V. Filipović, “Proposition for improvement tournament selection operator in genetic algorithms”, MS thesis, Faculty of Mathematics, Belgrade, 1998. (in Serbian).

[35] J. Grefenstette, “Virtual genetic algorithms: first results”, Internal report, NAVY Center for Applied Research in Artificial Intelligence (NCARAI), 1995.

[36] J. Heitkoetter, D. Beasley, “The Hitch-Hiker’s guide to evolutionary computation”, FAQ in comp.ai.genetic, 1999.

[37] K. Holmberg, “Experiments with primal-dual decomposition and subgradient methods for the uncapacitated facility location problem, Research Report LiTH – MAT / OPT - WP – 1995 - 08, Optimization, Department of Mathematics, Linkoping Institute of Technology, Sweden, 1995.

[38] A. Juels and M. Wattenberg, “Stochastic hill climbing as a baseline method for evaluating genetic algorithms”, Internal report, Groupe de Bio-Informatique, Ecole Normale Superieure, september 1994.

[39] J. Kratica, "Parallelization of genetic algorithms for solving some NP-complete problems", PhD thesis, Faculty of Mathematics, Belgrade, 2000. (in Serbian).

THE AUTHORS

Vladimir Filipović was born in Podgorica, Montenegro, Yugoslavia (1968). He received his B.S. degree in computer science (1993) and M.Sc. in computer science (1998) from University of Belgrade, Faculty of Mathematics. Since 1994 is teaching assistant at the Faculty of Mathematics. Since 1994 is teaching assistant at the Faculty of Mathematics. Also working as a software consultant for Analytx, Inc. USA. His research interests include genetic algorithms, parallel algorithms and operational research.

Jozef Kratica was born in 1966 in Belgrade, Serbia, Yugoslavia. He received his B.S. degrees in mathematics (1988) and computer science (1988), M.Sc. in mathematics (1994) and Ph.D. in computer science (2000) from University of Belgrade, Faculty of Mathematics. In 2000, he joined Mathematical Institute as a researcher. As a delegation leader participated on the International Olympiads in Informatics (IOI'90 Minsk - Belarus, IOI'93 Mendoza - Argentina). His research interests include genetic algorithms (evolutionary computation), parallel and distributed computing and location problems.

Dušan Tošić was born in Knjaževac, Serbia, Yugoslavia (1949). He received his B.S. degree in mathematics (1972), M.Sc. in mathematics (1978) and Ph.D. in mathematics (1984) from University of Belgrade, Faculty of Mathematics. Since 1985 he has been professor of computer science at Faculty of Mathematics His research interests include parallel algorithms, optimization, and evolutionary computation.

Ivana Ljubić was born in 1973 in Prokuplje, Serbia, Yugoslavia. She received his B.S. degree in computer science (1996) and M.Sc. in mathematics (2000) from University of Belgrade, Faculty of Mathematics. She currently working on Ph.D. studies at Vienna University of Technology**,** Institute for Computer Graphics. Her research interests include evolutionary computation and biconnectivity augmentation problems.