International Journal of Pure and Applied Mathematics
Volume 48 No. 2 2008, 1-16

A NEW GENETIC ALGORITHM APPLIED TO THE TRAVELING SALESMAN PROBLEM

Sawsan K. Amous[1], Taïcir Loukil[2], Semya Elaoud[3], Clarisse Dhaenens[4 §]
[1,2,3]Faculty of Economics and Management
University of Sfax
Sfax, TUNISIA
[1]e-mail: amoussawsan@yahoo.fr
[2]e-mail: taicir.loukil@fsegs.rnu.tn
[3]e-mail: samyaelaoud@yahoo.fr
[4]Laboratoire d'Informatique Fondamentale de Lille (LIFL)
Université des Sciences et Technologies de Lille
LIFL - UMR USTL/CNRS 8022 - Bâtiment, M3
Villeneuve d'Ascq Cédex, 59655, FRANCE
e-mail: Clarisse.Dhaenens@lifl.fr

Abstract: Genetic algorithms can be applied to a wide class of combinatorial optimization problems. In this paper, we propose an efficient genetic algorithm (GAs) with some innovative features to solve the traveling salesman problem. We propose a new mutation operator called "Mutation by Extended Elimination", a revised order Crossover and an Elite Selection Method. This algorithm is tested on a set of benchmarks from the TSP-LIB and compared with a previously published GA. The results show the efficiency of the algorithm when applied on both the Traveling Salesman Problem and a scheduling problem from the literature.

AMS Subject Classification: 90A08, 90BXX
Key Words: genetic algorithm, traveling salesman problem, combinatorial optimization problem

Received: March 13, 2008

§Correspondence author

1. Introduction

The Traveling Salesman Problem (TSP) can be stated as the problem of finding the shortest Hamiltonian tour, of K cities, which visits (each city is visited once and only once). In this work, we are interested in the standard (or symmetric) traveling salesman problem where the distance from i to j is the same than from j to i. The Traveling Salesman Problem is widely studied. As this problem is NP-hard, it is not possible (except if $P = NP$) to find a polynomial optimization method. Hence many approaches have been proposed: advanced exact methods in order to try to solve even larger problems and heuristics (and in particular metaheuristics) in order to solve some real size problems.

Genetic algorithms (GAs) are such metaheuristics. These stochastic methods have already been successively applied to solve a wide range of combinatorial optimization problems due to their ease of adaptability and applicability to the problem at hand. They distinguish themselves by their well structured problem description.

The remainder of this paper is organized as follows. First, an introduction to the TSP and aliterature review on solving the TSP using GAs is given. Then, we propose the modified GAs with a revised method of the Order Crossover adapted to the Traveling Salesman Problem, an "Elite Selection Method" and a new local research procedure for the mutation called "Mutation by Extended Elimination". Finally, a computational comparison of the various solution approaches is presented followed by concluding remarks and suggestions for future research in this area.

2. Literature Review

Several methods have been proposed for obtaining either optimal or near optimal solutions for the TSP. For a good overview of the TSP and various proposed solutions methodologies, see [10]. Metaheuristics have been generally applied to large scale problems, in particular GA. They have shown better performances when dialing with different TSP structures. Genetic algorithms strongly differ from other heuristics in conception; the basic difference is that while local search methods always process single points in the search space, genetic algorithms maintain a population of potential solutions and so, perform a multidirectional search [2]. The algorithm starts from an initial population of candidate solutions or individuals and proceeds for a certain number of iterations until one or more stopping criteria is (are) satisfied. This evolution is directed by a fitness measure function that assigns to each solution (represented by a chromosome) a quality value. Once the population is evaluated, the selection operator chooses which chromosomes in the population will be allowed to reproduce. The stronger an individual is, the greater chance of contributing to the production of new individuals it has. The new individuals inherit the properties of their parents and may be created by crossover (the probabilistic exchange of values between chromosomes) or mutation (the random replacement of values in a chromosome). Continuation of this process through a number of generations will result in a group of solutions with better fitness in which optimal or near-optimal solutions can be found.

Katayama et al [9] presented an efficient genetic algorithm for solving the traveling salesman problem as a combinatorial optimization problem. Carter et al [2] presented a new approach to solving the multiple traveling salesman problem using genetic algorithms. Louis et al [11] examined the feasibility of using GAs with a long term memory to attack similar TSP. Qu et al [17] developped a synergic approach to GAs for solving TSP. They study some typical self-organizing behavior exhibited in GAs for solving TSP. These behaviors include the exponential relationship, entropy jumping phenomenon, assimilation and entropy synchronization and they propose to use "doping" as the measure to prevent the premature convergence indexing of the GAs. GA was also presented and implemented on a cluster of workstations by Sena et al [21].

We can combine GAs in order to attempt high quality solutions particularity for large problems instances. For example, Bui et al [1] combined a local search method with GAs for the TSP. Schleuter et al [19] have proposed a GA where all individuals of the population are local minima with respect to the embedded local search method. Merz and Freisleben[14] proposed new operators for Global Local Search which are designed to produce better individuals from existing ones with the ability to find optimal solution for symmetric TSP instances of up to 1400 cities. Xiulan et al [22] presented an effective genetic algorithm that is implemented in real-code and only blend crossover operators are applied to two randomly selected individuals from the existing population.

The key to find a good solution using a GA lies in developing a good chromosome or mutation representation of solutions to the problem. The development of effective GA operators for TSP led to a great deal of interest and research to improve the performance of GAs for this type of problem (see [15], [17] and [9]). Several summaries of solving TSP with GAs have been published. Comprehensive reviews of the operators and associated issues were provided (see [16], [20] and [11]). Classical GAs operators produce redundant solutions hence, a well-designed chromosome should reduce or eliminate redundancy. Therefore, we propose to modify the classical genetic operators, for more diversification and intensification, in order to find a better near optimal solution.

3. Modified Genetic Operators

Solving the TSP using GAs has generated a great deal of research on how best to perform the action of "evolving" an optimal (or good) solution to the problem [2]. Intensification and diversification are two important factors in the assessment of the GAs process. The intensification is insured by the selection process while mutation and crossover operators are means of diversification [6]. A good compromise should be found to propose an efficient GAs. Selection, crossover and mutation procedures should work synergistically to guide the search process and to adjust the balance between diversification and intensification. We propose in this paragraph three genetic operators with some innovative features: an "Elite Selection Method", a revised method of the Order Crossover adapted to the Traveling Salesman Problem, and a new local research procedure for the mutation.

3.1. The Elite Selection Method Review

Selection is one of the main used operators in evolutionary algorithms and its primary objective is to emphasize better solutions of a population [5]. It consists in choosing n parents from N individuals to participate in the production of offspring for the next generation, considering their fitness [7]: individuals with better fitness values are picked more frequently than individuals with worse fitness values. Our selection method is not based on a probabilistic random process using the strength of the solution (as it is in the roulette wheel selection). It consists in sorting out the population from the most effective to the least effective and only the best solutions will be used for the crossover

$$n = E(\sqrt{N})\,, \tag{1}$$

where: N represents the population size; $E()$ – the largest integer part; n – the number of solutions allowed to reproduce (number of solutions in the matting pool).

Then, we have exactly C_n^2 different parent pairs and, consequently, C_n^2 different children. To keep an unchanged population size, some additional randomly selected parents n' from the n solutions of the current population are directly copied into the next generation (so considered as new children). So n' is the difference between the fixed population size and the number of resulting children

$$n' = N - C_n^2\,. \tag{2}$$

Let us notice that, the use of only powerful parents in the reproduction process may lead to a high level of intensification. Therefore, we used a high crossover probability and a specified crossover operator. Hence, we present in the next section a revised order crossover in which we added some diversity features.

3.2. Revised Order Crossover (ROX)

Recombination is a process in which new individuals are generated by exchanging features of the selected parents with the intent of improving the fitness of the next generation. Since the best features are not known a priori, individuals are generally recombined by randomly exchanging subparts of their parents. The new strings have new characteristics and will be added to the population. One of the most efficient crossovers adapted to the TSP is the Order Crossover (OX) (see [4] and [8]). It creates new offspring by choosing a sub-tour of one parent and preserving the relative order of cities of the other parent. Let us consider in figure 1, two parents tours with two cut points marked by the symbol |.

Parent 1 :	1	2 |	3	4	5 |	6	7	8
Parent 2 :	6	7 |	4	2	8 |	5	3	1

Figure 1: Example of parents

The offspring are constructed in the following way. First, the tour subsequences between the cut points are inherited into the offspring, which is shown Figure 2.

Offspring 1 :	*	*	3	4	5	*	*	*
Offspring 2 :	*	*	4	2	8	*	*	*

Figure 2: Subsequences of offspring

Second, delete the cities, which are already present in the subsequence from the other parent (Figure 3).

Offspring 1	:	$*$	$*$	3	4	5	$*$	$*$	$*$
Parent tour 2	:	6	7	4	2	8	5	3	1
Offspring 2	:	$*$	$*$	4	2	8	$*$	$*$	$*$
Parent tour 1	:	1	2	3	4	5	6	7	8

Figure 3: The order crossover process

Then write down the genes from each parent chromosome starting from the second crossover point. So we obtain in Figure 4 the new offspring.

Offspring 1 :	1	6	3	4	5	7	2	8
Offspring 2 :	6	7	4	2	8	1	3	5

Figure 4: Offspring after order crossover

Attracted by the efficiency of OX, we propose to modify this operator including more diversification features in order to cope with the high intensification level of the proposed selection.

Revised Order Crossover (ROX) mainly preserves the original OX principle. A new offspring is created by randomly choosing a sub-tour of one parent (see Figure 5). Genes of the first parent are copied in the generation of the second offspring and vice versa. After that, we respectively copy in the first offspring the genes of the first parent and we do not copy the ones that already exist. We repeat the same procedure with the second offspring. Figures 5 to 7 illustrate the way ROX operates. First, we consider the following parent tours with two cut points designed by the symbol | (Figure 5).

Parent 1 :	1	2 |	3	4	5 |	6	7	8
Parent 2 :	6	7 |	4	2	8 |	5	3	1

Figure 5: Randomly choosing a sub-tour of parents

Secondly, offspring inherit the sub-tours of parents (Figure 6).

After that, remaining genes are used to complete chromosome. The completion of offspring one is made, starting from the first parent and the first place. Genes that were between the crossovers points are deleted (see Figure 7). The construction of offspring 2 is made in a similar way.

Offspring 1 :	$*$	$*$	4	2	8	$*$	$*$	$*$
Offspring 2 :	$*$	$*$	3	4	5	$*$	$*$	$*$

Figure 6: Switching the genes

Offspring 1 :	1	3	4	2	8	5	6	7
Offspring 2 :	6	7	3	4	5	2	8	1

Figure 7: Offspring after revised order crossover

Doing crossover by just interchanging any parts of parents, may lead to premature convergence of the algorithm since in same cases no changes will be made on the composition of any parent, perhaps just through mutation. That is why we propose to modify the OX and we introduce some diversification features in order to cope with the whole proposed algorithm characteristics and the problem at hand.

3.3. Mutation by Extended Elimination (MEE)

The mutation procedure is a noisy procedure which modifies each individual independently. It aims to keep diversity in the population and promotes the search in the solution space due to its ability of rapidly generating new building blocks [12]. This genetic operator has a significant effect on the performance of the algorithm. However, a pure random choice of genes on which the transformation is done may direct the search towards undesirable search regions. Therefore, incorporating problem specific knowledge into this operator is of greet importance on the evolution process.

The mutation by extended elimination (MEE) searches for the highest cost between two successive cities. These cites are then, exchanged with their neighbors. If one of these chosen chromosomes comes in an extreme position, it will be reversed with the one in the other extreme position, i.e., the chromosome in the last position, will be reversed by the one in the first position and vice versa. As an example, if the highest cost is between cities 8 and 5, in offspring 1, cities 8 and 5 will be switched with their neighbors 2 and 6 respectively (see Figure 8).

The MEE procedure promotes better search regions. This operator is adapted to the TSP because it eliminates the highest costs and helps avoid-

Offspring 1 :	1	3	4	2	**8**	**5**	6	7
Offspring 1 :	1	3	4	**8**	**2**	**6**	**5**	7

Figure 8: The solution after mutation

ing the construction of unsuitable offspring.

4. Computational Experiments

The genetic algorithm is characterized by two fundamental, dichotomous forces competing within the evolution of the population: exploitation and exploration. The selection operator exploits the current knowledge of the solution space by propagating the better guesses and discarding the poorer ones. The crossover and mutation operators explore the search space by creating new guesses. The balance is adjusted by changing the relative probabilities.

The genetic algorithm developed in our research has been programmed with *Visual Basic C++* on a *Pentium 4*, 1.7MHz machine. Every solution is a permutation between 1 and N (N is the total number of cities). The objective is to minimize the total distance (or time). For each experiment, 10 tests (the average is taken) have been executed.

4.1. Solving a Scheduling Problem

As operators proposed for the TSP problem may be used for any permutation problem, we first evaluate the practical benefits of the proposed algorithms (MOX-MEE), by comparing results obtained on a scheduling example of the literatures [13]. In this article, a genetic algorithm is applied to solve a sequence dependent changeover times on a single machine (15 jobs). As this scheduling problem may be modeled as a TSP problem, classical approaches for TSP may be applied to this problem. The best sequence found with the ROX is the following (14 - 12 - 6 - 9 - 1 - 5 - 2 - 8 - 3 - 7 - 10 - 13 - 4 - 15 - 11), with a cost of 47,5. The best sequence starting with the machine 14, found by [13], is (14 - 9 - 1 - 12 - 6 - 11 - 15 - 5 - 3 - 8 - 13 - 4 - 10 - 7 - 2) with a cost of 48,9. The results from the tests show the efficiency of our proposed algorithm while

Problem	Optimal	OX - OM	ROX - OM	OX - MEE	ROX - MEE
Bayg 29	1610	1610	1610	1610	1610
Bays 29	2020	2022	2025	2020	2020
Eil 51	426	485	502	463	455
Berlin 52	7542	8189	8157	7542	7542
Eil 76	538	584	571	550	557
Eil 101	629	665	672	667	633
Rat 195	2323	3415	3236	3064	2812

Table 1: Comparison of configurations

starting with the same job.

4.2. Comparison of Operators

For comparative purpose, four tests are run with different operators' combinations in order to assess their efficiency. Benchmark instances taken from the TSPLIB library are used [18]. In first time, we applied the order crossover of Davis [4] and the roulette wheel selection (OM), in this method the selection probability of each individual is calculated by dividing its fitness by the sum of the fitnesses of all individuals. In second time, we applied the mutation by extended elimination (MEE) instead of the (OM). After that, the crossover is changed by our proposed (ROX), and we change both operators (ROX-MEE). Combinations tested are:

— The Order Crossover (OX) and the Ordinary Mutation (OM: roulette);

— The Order Crossover (OX) and the Mutation by Extended Elimination (MEE);

— The Revised Order Crossover (ROX) and the Ordinary Mutation (OM);

— The Revised Order Crossover (ROX) and the Mutation by Extended Elimination (MEE).

Results are resumed in Table 1.

Table 1 reports the average solution value for different problems. It also indicates, for the comparison, the optimal value. This table shows that our algorithm (ROX-MEE) gives optimal or near optimal results for several instances. Another important aspect to see is the robustness of the algorithm over all the instances. Figure 9 resumes different deviation rates for the selected instances tested with the four configurations. For example, ROX- OM for the instances Eil 101 gives a deviation of about 8 percent from optimal when the deviation of ROX-MEE is only 2 percents.

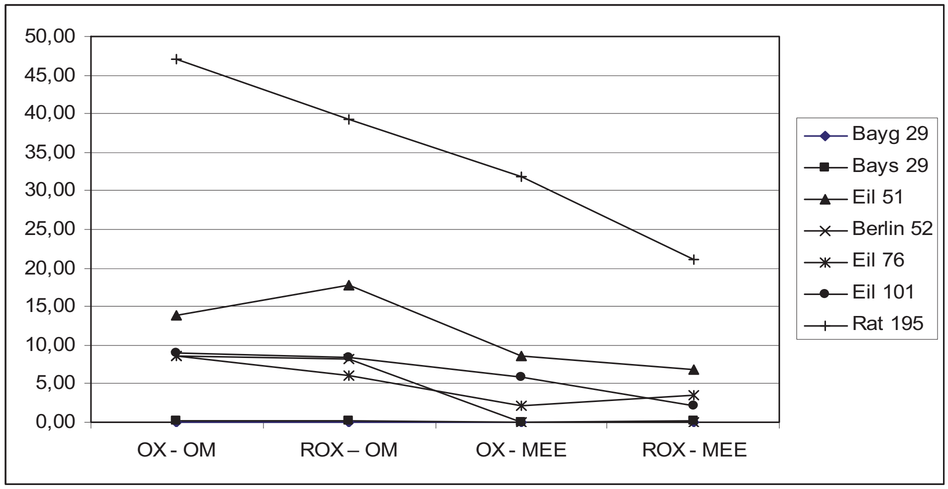

Figure 9: Comparison of different crossover operators

The quality of the results of our algorithm seems satisfying. However, as the number of cities increases (and the solution space grows), the ROX-MEE begins to exhibit a less efficiency. While the objective of minimizing the total distance traveled is interesting, we repeated our tests using MOX and MEE for others instances. The results of these tests are presented in Table 2.

In order to deeply test and compare the performance of the proposed operators, computational experiments were performed with 12 instances. These instances were classified with types and average of deviation of optimal (ADO). As we could expect, the run time (in seconds) grows up with the number of cities and the ADO diverges from optimal However, qualities of solutions obtained are still good. Other comparative tests were run. Chatterjee et al [3] proposed a genetic algorithm to solve TSP but the deviation from the optimal is between 1.30 percents and 2.10 percents and the algorithm found the result after one hour to small benchmarks instances.

Problem	Optimal	Types	ADO*	Iterations	Generations	Times
Bayg 29	1610	GEO	0	1000	200	0
Bays 29	2020	GEO	0	1000	500	1
Berlin 52	7542	EUD-2D	0	1000	200	0.2
Eil 51	426	EUD-2D	0.06	1000	200	0.2
Rat 99	1211	EUD-2D	0.02	1000	500	1.0
Eil 101	629	EUD-2D	0.63	1000	200	1.2
Gr 24	1272	MATRIX	0.003	1000	500	0.5
KroA 100	21282	EUD-2D	0.2	1000	200	1.6
Ch 130	6110	EUD-2D	0.33	1000	200	1.6
Brg 180	1950	MATRIX	0.89	1000	300	2.9
pr1002	259045	EUD-2D	2.65	1000	500	45.5

*Average Deviation of the Optimal $(ADO) = \frac{Fitness-Optimal}{Optimal}$

Table 2: ROX-MEE tests

Figure 10 shows that the results of the ROX-MEE algorithm are very close to the optimal ones and even optimal for the first four instances. For other evaluations we used the well known standard benchmark set and we chose 30 different problem instances ranging from 22 cities to 1002 which results are summarized in Table 3. Each benchmark is run 10 times and we take the minimum result, the maximum, the variance, the gaps and we average the results for each considered instances.

Finally, it can be seen from Table 3 the efficiency of ROX-MEE algorithm for the TSP and we conclude that the best results obtained by our algorithm are very close to the optimal comparing with other methods. Meanwhile, it can be seen that the difference between optimal and minimum values is very small when the size is limited, this efficiency can be considered to be the mutual result of the properties of the GA using both the proposed crossover and MEE which also shows the robustness of the algorithm.

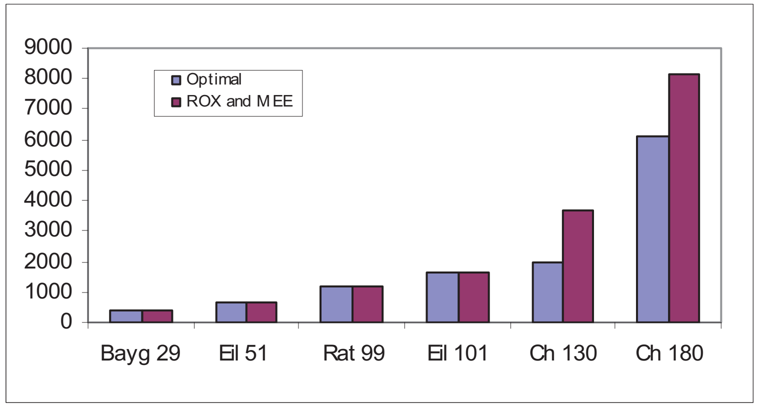

Figure 10: Comparison of optimal and ROX-MEE

5. Conclusion

Genetic algorithms appear to find good solutions for the traveling salesman problem; however it depends very much on the way the problem is encoded and the choice of operators (crossover and mutation methods). It seems that the methods that use heuristic information or encode the edges of the tour perform the best and give good indications for future work in this area. As yet, genetic algorithms have not found a better solution to the traveling salesman problem than is already known, but many of the already known best solutions have been found by some genetic algorithm method also.

In this paper we propose, new genetic operators adapted to the traveling salesman problem. We propose an elite section method, a revised order crossover and a mutation by extend elimination. The proposed algorithm was compared with a previous result from literature and with some benchmark problems too.

Results show that modeling the TSP using the new genetic operators proposed has clear advantages over using one of the classical operators. The results indicate that the improved algorithm is able to get good solutions when tested on a scheduling problem coming from the literature and benchmark instances taken from the TSP library. The ROX-MEE has performed well in theoretical and empirical comparisons. However, when the number of cities increased (and the solution space grows), the average time increases too.

The proposed algorithm seems to be a promising approach. An interest-

Name	Types	Opt.	Min	Max	Avg.	Var.	Gap	Time
A 280	EUD-2D	2579	2580	3040	2752.2	26300	0,0626	30
Att 48	ATT	10628	10751	13200	11651	616105	0.0773	6
Bayg 29	GEO	1610	1610	1820	1681	4709	0,0422	0
Bays 29	GEO	2020	2020	2100	2041,1	618.86	0,0103	0
Berlin 52	EUD-2D	7542	7690	8716	8221,8	118839	0,0647	1
Brazil58	MATRIX	25395	28717	36258	31362	6E+0.6	0,0843	5
Brg 180	UPPER-ROW	1950	1980	2314	2086,7	12630	0,0511	15
Burma 14	GEO	3323	3743	4500	4159	58721	0.1	1
Ch 130	EUD-2D	6110	6110	6540	6343,4	17426	0,0368	13
Ch 150	EUD-2D	6528	6590	7012	6844,1	24233	0.0372	16
Eil 101	EUD-2D	629	641	785	696.9	2048.9	0,0802	10
Eil 51	EUD-2D	426	426	485	440,3	445,41	0,0325	3
Eil 76	EUD-2D	538	538	550	543.2	15.56	0,0096	5
Gil 262	EUD-2D	2378	2540	4010	3331,8	177567	0,2376	142
Gr 202	GEO	40160	41160	48761	44430,1	4884988	0,0736	187
Gr 96	GEO	55209	59764	66570	61964	4E+06	0,0355	96
KroA100	EUD-2D	21282	22282	29147	25398	5E+06	0,1227	45
KroB100	EUD-2D	22141	24684	32150	29002	9E+06	0,1489	55
KroC100	EUD-2D	20749	22210	23749	22818	233121	0,0266	42
KroD100	EUD-2D	21294	22185	24120	23111	437157	0,0401	53
Lin 105	EUD-2D	14379	15037	17240	15981	379043	0,0591	71
Lin 318	EUD-2D	42029	45030	48698	46585	1E+06	0,0334	283

Table 3: Benchmarks instances tests

ing extension of this work will concern its application on the multiobjective Traveling Salesman Problems.

Name	Types	Opt.	Min	Max	Avg.	Var.	Gap	Time
Pr 76	EUD-2D	108159	151295	173553	163624	5E+07	0,0754	121
pr1002	EUD-2D	259045	671288	882855	760262	5E+09	0,117	899
Rat 575	EUD-2D	6773	6761	7733	7257,6	98022	0,0684	110
Rat 783	EUD-2D	8806	9321	11023	10249	249571	0,0906	268
Rat 99	EUD-2D	1211	1211	2030	1502,6	85593	0,1941	11
Ts 225	EUD-2D	126643	196341	371254	269472	4E+09	0,2714	317
Tsp 225	EUD-2D	3919	4321	5346	4826,2	99397	0,1047	211
Ulysses22	GEO	7013	7013	7196	7063.5	2987.1	0.0071	1

Table 3: Continuation: Benchmarks instances tests

References

[1] T.G. Bui, B. Moon, A new genetic approach for the traveling salesman problem, In: *Proceedings of the First IEEE International Conference on Evolutionary Computation* (1994), 7-12.

[2] A.E. Carter, C.T. Ragsdale, A new approach to solving the multiple traveling salesperson problem using genetic algorithms, *European Journal of Operational Research*, **175** (2006), 246-257.

[3] S. Chatterjee, C. Carrera, L. Lynch, Genetic algorithms and traveling salesman problems, *European Journal of Operational Research*, **93**, No. 3 (1996), 490-510.

[4] L. Davis, Job shop scheduling with genetic algorithms, In: *Proceedings of the International Conference on Genetic Algorithms*, London (1985), 136-140.

[5] K. Deb, Introduction to selection, *Evolutionary Computation 1: Basic Algorithms and Operators* (2000), 166-171.

[6] S. Elaoud, J. Teghem, B. Bouaziz, Genetic algorithms to solve the cover printing problem, *Computers and Operations Research*, **34**, No. 11 (2007), 3346-3361.

[7] C.M. Fonseca, P.J. Fleming, Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization, In: *Proceedings of the Fifth International Conference on Genetic Algorithms* (1993), 416-423.

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

[9] K. Katayama, H. Sakamoto, H. Narihisa, The efficiencies of hybrid Mutation Genetic Algorithm for the TSP, *Mathematical and Computer Modeling*, **31** (2000), 197-203.

[10] E.L. Lawler, J.K. Lenstra, K. A. Rinnooy, D. Shimoys, *The Traveling Salesman Problem*, John Wiley and Sons, New York (1985).

[11] S.J. Louis, L. Gang, Case injected genetic algorithms for traveling salesman problems, *Information Science*, **122** (2000), 201-225.

[12] H. Lu, G.G. Yen, Rank-density-based multiobjective genetic algorithm and benchmark test function study, *IEEE Transactions on Evolutionary Computation*, **7**, No. 4 (2003), 325-343.

[13] C.J. Malmborg, A genetic algorithm for service level based vehicle scheduling, *European Journal of Operational Research*, **93**, No. 1 (1996), 121-134.

[14] P. Merz, B. Freisleben, A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems, In: *Proceeding of the 1996 IEEE International Conference on Evolutionary Computation*, Japan (1996), 616-621.

[15] P.W. Poon, J.N. Carter, Genetic algorithm crossover operators for ordering applications, *Computers and Operations Research*, **22**, No. 1 (1995), 135-147.

[16] J. Potvin, Genetic algorithms for the traveling salesman problems, *Annals of Operations Research*, **63** (1996), 330-370.

[17] L. Qu, R. Sun, A synergetic approach to genetic algorithms for solving traveling salesman problem, *Information Sciences*, **117**, No-s: 3, 4 (1999), 267-283.

[18] G. Reinelt, TSPLIB. University of Heidelberg, http://www.iwr.uni-heidelberg.de/iwr/compot/soft/TSPLIB95/TSPLIB.html (1996).

[19] J.D. Schleuter, L.J. Eshelman, D. Offutt, Spurious correlations and premature convergence in genetic algorithms, In: *Foundations of Genetic Algorithms*, Morgan Kaugmann Publishers (Ed. G.J.E. Rawlings), San Mateo, CA (1995), 102-112.

[20] L. Schmitt, M. Amini, Performance characteristics of alternative genetic algorithmic approaches to the traveling salesman problem using path representation: An empirical study, *European Journal of Operational Research*, **108**, No. 3 (1998), 551-570.

[21] G.A. Sena, D. Meghabi, G. Isern, Implementation of a parallel genetic algorithm on a culver of works talons: Traveling salesman problem, a case study, *FGCS*, **17** (2001), 477-488.

[22] W. Xiulan, X. Qingguan, Z. Yibing, An effective genetic algorithm for circularity error unified evaluation, *International Journal of Machine Tools and Manufacture*, **46** (2006), 1770-1777.