Expert Systems with Applications 38 (2011) 1313–1320

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Development a new mutation operator to solve the Traveling Salesman Problem by aid of Genetic Algorithms

Murat Albayrak [a], Novruz Allahverdi [b,*]

[a] Center Primary Education Scholl, Selki, Huyuk, Konya, Turkey
[b] Electronic and Computer Education Department, Technical Education Faculty, Selcuk University, Konya, Turkey

ARTICLE INFO

Keywords:
Genetic Algorithm
Mutation operator
Greedy methods
Traveling Salesman Problem
Optimization

ABSTRACT

In this study, a new mutation operator has been developed to increase Genetic Algorithm (GA) performance to find the shortest distance in the known Traveling Salesman Problem (TSP). We called this method as Greedy Sub Tour Mutation (GSTM). There exist two different greedy search methods and a component that provides a distortion in this new operator. The developed GSTM operator was tested with simple GA mutation operators in 14 different TSP examples selected from TSPLIB. The application of this GSTM operator gives much more effective results regarding to the best and average error values. The GSTM operator used with simple GAs decreases the best error values according to the other mutation operators with the ratio of between 74.24% and 88.32% and average error values between 59.42% and 79.51%.

1. Introduction

Traveling Salesman Problem (TSP) is most widely studied combinational optimization problem (Helsgaun, 2000). TSP is based upon the defining of a salesman's closed shortest tour that will be between n cities. The salesman has to visit every city once and return to the starting city in a shortest way. If ($dist(c_i, c_j)$) represents the distance between traveled cities and the coordinates (c_ix_i, c_iy) of all cities (c_i) that the salesman will visit is known, then total distance of implemented tour (f) can be described by;

$$f = \sum_{i=1}^{n-1} \sqrt{(c_i x - c_{i+1} x)^2 + (c_i y - c_{i+1} y)^2} + \sqrt{(c_n \cdot x - c_1 \cdot x)^2 + (c_n y - c_1 y)^2} \quad (1)$$

If a distance matrix ($d(i,j)$) which contains a distance from a city (c_i), to another city (c_j) taken place in TSP is formed, then length (f) of walking tour (T) expressed as follows:

$$f = \sum_{i=1}^{n-1} d(T[i], T[i+1]) + d(T[n], T[1]) \quad (2)$$

Performance of TSP resolving process is evaluated by taking into account two parameters. These parameters are solution time and error value. Nowadays for TSP the solutions that have several percentage low error value and are produced quickly can be accepted as approximate. There exist some intuitive methods for finding approximate solutions (Dorigo & Gambardella, 1997; Jayalakshmi, Sathiamoorthy, & Rajaram, 2001; Misevicius, 2004; Schleuter, 1997; Seo & Moon, 2002; Stützle & Hoos, 1997). In contrast to exact algorithms the *approximate* algorithms obtain good solutions but do not guarantee for finding optimal solutions. These algorithms are usually very simple and have relatively short running times. Some of the algorithms give solutions that differ in average only by a few percent from the optimal solution. Therefore, if a small deviation from optimum can be accepted, it may be appropriate to use an approximate algorithm (Albayrak, 2008; Helsgaun, 2000).

In this study, a new mutation operator has been developed to increase application performance of GA on TSP and this operator was compared with other GA mutation operators. The rest of the paper is organized as follows: in Section 2, GA is examined and related background theory is presented. In Section 3, developed Greedy Sub Tour Mutation (GSTM) operator is described. In Section 4, experimental results are evaluated. In the final, this study is concluded.

2. Genetic Algorithms

Genetic Algorithm (GA) that is developed by Holland (1975), is a type of directed random search algorithm and recursive search technique based on natural election used in computing to find approximate solutions for optimization and search problems (Genetic Algorithms, 2009; Holland, 1975). Main aim of GA is to achieve better results by removing bad results during production of population from current generation to next generation and using

* Corresponding author. Tel.: + 90 332 2233356.
E-mail address: noval@selcuk.edu.tr (N. Allahverdi).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.07.006

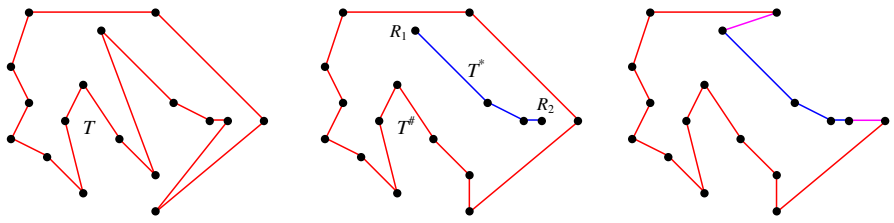
```
Procedure GA;
Begin
        Produce initial population P that includes randomly produced solutions;
        Repeat
        Find the fitness of individuals in population;
        Select parents for the operation of reproduction (i_a, i_b ∈ P);
    Reproduce new individual by applying crossover operation on parents with
    probability rate of P_C;
        Apply mutation operator on new individual with probability rate of P_M;
        Add the newly generated individual to the population;
        Exchange new population with the previous population;
    Until stopping condition is obtained;
End
```

Fig. 1. A simple Genetic Algorithm.

```
P_RC     : Reconnection Probability
P_CP     : Correction and Perturbation Probability
P_L      : Linearity Probability
L_MIN    : Minimum Sub Tour Length
L_MAX    : Maximum Sub Tour Length
NL_MAX   : Neighborhood List Size

Step 1: Define randomly starting (R1) and ending points (R2) of sub-tour;
Step 2: According to the random (Rnd) number produced between (0–1);
If (Rnd≤P_RC) then,
{
        Subtract [R1-R2] sub-tour from tour  T ( T# ← T – [R1-R2] ; T* ← [R1-R2] );
        Hold on T* sub-tour that causes minimum extension to T# tour;
}Else{
        According to the random (Rnd) number produced between (0–1);;
        If (Rnd≤P_CP) then
        {
                Copy [R1-R2] sub-tour in the T tour (T* ← [R1-R2]);
                Add each element of T* sub-tour to the T tour starting from the position
        R1 by rolling or mixing with P_L probability;
        }Else
        {
                Select randomly one neighbor from neighbor lists for the points R1 and
        R2 (NL_R1 and NL_R2);
        Invert the points NL_R1 or NL_R2 that provides maximum gain such a way that
these points will be neighbors to the points R1 or R2;
        Repeat if the inversion is not taken place;
        }
}
```

Fig. 2. Greedy Sub Tour Mutation (GSTM) algorithm.

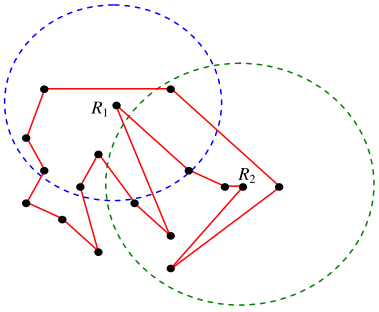


Original T tour

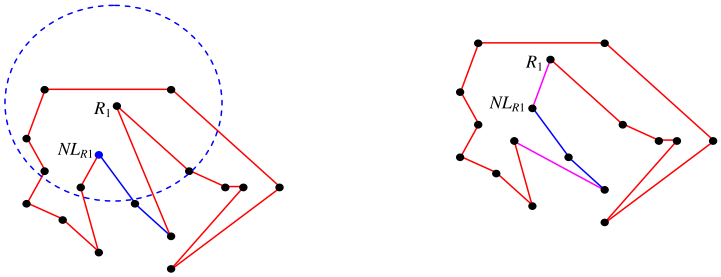
T* sub-tour subtracts from the T tour

T^* sub-tour connects to the $T^\#$ tour by the way where $T^\#$ tour will extend minimum

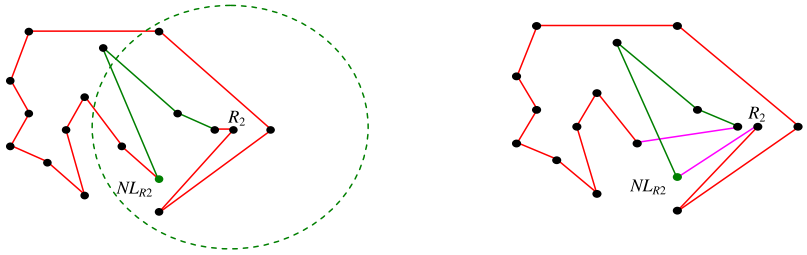
Fig. 3. Greedy reconnection.



Nearest neighbors of the cities R_1 and R_2 selected randomly for sub tour (NL_{MAX}=7)



Rotation of randomly selected city NL_{R1} from the neighbor list of R_1 city by the way as it will a neighbor to the city R_1 Gaining (G_{R1}) = $d[T[R_1], T[R_1+1]] + d[T[NL_{R1}], T[NL_{R1}+1]] - (d[T[R_1], T[NL_{R1}]] + d[T[R_1+1], T[NL_{R1}+1]])$



Rotation of randomly selected city NL_{R2} from the neighbor list of R_2 city by the way as it will a neighbor to the city R_2 Gaining (G_{R2}) = $d[T[R_2], T[R_2+1]] + d[T[NL_{R2}], T[NL_{R2}+1]] - (d[T[R_2], T[NL_{R2}]] + d[T[R_2+1], T[NL_{R2}+1]])$

Fig. 4. Sub tour rotation on neighbor lists.

$T^* \leftarrow T[R_1\text{-}R_2]$
$w = R_1$
Repeat next operations until added cities will not remain in tour T^* {
  According to the randomly produced number (Rnd) between (0-1);
  If ($Rnd \leq P_L$)
  {
    Then add a random city from the tour T^* to the w position of tour;
  } else
  {
    Add the last city of the tour T^* to the position w of the tour T
  }
  Subtract added city in the tour T from the tour T^*;
  $w = w + 1$;
}

Fig. 5. Algorithm of the distortion method.

only good results to achieve the better results. Here, the fitness function ($f(x)$) that defines what is the "better" has to be prepared very carefully depending on the problem.

Possible solutions in GA are presented by chromosomes and generally the first solutions are produced randomly. Chromosomes (individuals) together generate a set of solution populations. The fitness function formed properly to the problem, presents a solution quality of the individuals. To produce new and good solutions, GA uses operators such as selection, crossover and mutation. To produce new generation crossover and mutation operators are applied on the two individuals that are selected from population by selection mechanism. The basic steps to follow for solution of a problem by GA are shown in Fig. 1.

Table 1
Analysis-1 of mutation methods.

Mutation methods	Problem	BERLIN52	KROA100	PR144	CH150	KROB150	PR152	RAT195	D198	KROA200	TS225	PR226	PR299	LIN318	PCB442	Average
EXC	Best Err. (%)	**0.0000**	0.1692	0.2426	1.9761	2.9277	2.0358	4.4770	1.9709	2.5061	3.0424	4.9186	10.2198	10.1026	7.3437	3.70945
	Ave. Err. (%)	2.0353	2.8714	**0.5330**	2.8232	5.0919	3.9900	5.9836	4.9728	4.9877	3.8174	7.2295	15.1775	11.7195	10.4774	5.83645
	Ave. time (ms)	1820	7004	9686	9720	14,387	7990	8154	11,345	15,646	6804	7518	11,203	14,317	10,836	9745
DISP	Best Err. (%)	**0.0663**	4.2947	1.4845	3.4161	7.8263	4.7216	5.0366	3.8910	6.7284	3.3693	4.1322	12.2616	10.9996	11.3435	5.68369
	Ave. Err. (%)	1.4240	8.0552	1.7628	3.9859	10.4635	5.5932	6.9393	5.9835	8.8276	3.8916	7.9930	16.8490	12.9013	12.3053	7.64109
	Ave. time (ms)	2385	**5354**	**4212**	4689	**5962**	**4381**	**4956**	**4781**	7171	6110	6142	**7395**	**5989**	**5968**	**5393**
INV	Best Err. (%)	**0.0000**	2.6125	1.0284	2.9412	7.1183	3.1202	3.7021	2.6743	6.2245	3.1640	8.1512	10.9543	9.5410	10.5124	5.12460
	Ave. Err. (%)	1.1854	4.7965	1.9755	3.6596	9.0521	5.0318	6.3108	4.9823	8.3594	3.7355	8.8249	15.7060	13.0386	11.6322	7.02076
	Ave. time (ms)	3537	9335	4262	**4510**	7843	7221	8181	8464	**6017**	7090	**5906**	9092	8015	9757	7088
INS	Best Err. (%)	**0.0000**	0.4652	0.1879	0.6434	1.8178	1.3938	4.2617	2.0279	2.1554	3.1751	3.8423	3.8804	6.3099	8.9685	2.79495
	Ave. Err. (%)	0.1525	3.0726	1.3802	2.3943	4.5488	2.8468	5.7426	3.4100	4.2819	3.8767	6.4840	11.4187	10.3935	11.0585	5.07578
	Ave. time (ms)	2051	11,026	7000	9401	16,720	9225	10,000	17,110	15,629	6720	12,218	18,776	16,731	9445	11,575
SIM	Best Err. (%)	**0.0000**	0.1692	0.2255	0.8425	2.8397	1.5282	2.3676	1.6984	1.5766	1.6930	2.5769	6.4390	6.3932	6.3728	2.48017
	Ave. Err. (%)	0.6099	1.3114	1.1717	1.5809	3.8596	2.7976	4.1886	2.5241	3.2457	2.5818	4.1795	9.1264	8.7140	8.1240	3.85822
	Ave. time (ms)	2706	10,312	7542	11,804	17,456	10,742	11,793	18,498	15,117	10,701	12,995	20,093	16,042	18,735	13,181
SCM	Best Err. (%)	**0.0000**	0.8176	0.1589	0.9344	3.2836	1.5146	4.6492	3.1179	3.9873	2.7945	3.7875	10.8776	9.1651	7.8597	3.78199
	Ave. Err. (%)	**0.0000**	3.6966	1.3016	2.1078	7.0609	2.6940	5.9923	4.5818	5.4502	3.8029	5.0694	13.8686	10.4456	10.6251	5.47832
	Ave. time (ms)	1274	7381	8415	8234	9156	10,218	6226	11,201	10,076	5512	8617	14,304	11,135	10,092	8703
GSM	Best Err. (%)	**0.0000**	0.3007	0.2426	1.6544	3.4520	2.2977	4.3048	3.0482	4.4675	3.0424	5.0517	8.8191	9.4126	5.7643	3.70413
	Ave. Err. (%)	0.9931	3.6980	0.6891	2.4908	4.9541	2.9112	6.5476	4.2421	5.7018	4.4254	6.4485	13.6295	11.0036	8.0389	5.41240
	Ave. time (ms)	2423	5657	6661	9879	7937	8048	7001	10,921	10,440	**5359**	8848	13,690	13,139	13,671	8834
GSTM	Best Err. (%)	**0.0000**	**0.0000**	**0.0000**	**0.4596**	**0.9644**	**0.7695**	**0.6027**	**0.3866**	**0.8683**	**0.2527**	**0.7242**	**1.2326**	**0.9827**	**2.0501**	**0.66380**
	Ave. Err. (%)	**0.0000**	**1.1836**	1.0809	**0.6357**	**1.7616**	**1.6202**	**1.8425**	**1.2193**	**1.5432**	**0.4994**	**1.5287**	**2.9169**	**3.3099**	**2.7758**	**1.56554**
	Ave. time (ms)	**836**	6987	13,598	11,240	11,684	7937	15,050	12,096	13,292	11,559	13,843	17,424	14,643	19,132	12,094

Table 2
Analysis-2 of mutation methods.

Mutation methods	Problem Maximum running time for all trials (ms)	berlin52 836	kroA100 6987	pr144 13,598	ch150 11,240	kroB150 11,684	pr152 7937	rat195 15,050	d198 12,096	kroA200 13,292	ts225 11,559	pr226 13,843	pr299 17,424	lin318 14,643	pcb442 19,132	Average
EXC	Best Err. (%)	**0.0000**	0.3007	0.2426	0.9957	5.0785	1.3260	2.5829	2.8517	2.5402	3.1380	3.2463	10.5974	10.5546	9.3032	3.76841
	Ave. Err. (%)	1.8735	3.3136	0.6835	2.5215	7.1875	2.4226	5.3638	5.3213	5.7484	3.8766	6.4033	12.4002	11.8061	10.3366	5.66132
	Ave. time (ms)	825	7000	13,609	11,240	11,687	7937	15,062	12,109	13,297	11,562	13,845	17,437	14,656	19,156	12,102
DISP	Best Err. (%)	0.0663	5.0136	1.6878	2.8799	5.6640	3.7268	4.6492	4.6388	8.2130	3.2935	6.5361	12.1309	10.2524	10.5932	5.66754
	Ave. Err. (%)	1.9120	7.7070	1.7363	3.7163	8.8802	5.6464	6.3539	6.8308	9.0500	4.0203	9.0718	16.4080	11.9153	11.8717	7.50857
	Ave. time (ms)	843	7000	13,611	11,240	11,687	7939	15,064	12,109	13,297	11,565	13,843	17,439	14,661	19,156	**12,104**
INV	Best Err. (%)	0.0663	2.3400	1.0284	2.3744	6.0582	3.2070	2.6259	2.8834	4.7569	2.5513	5.7983	11.8176	9.2151	9.0275	4.55358
	Ave. Err. (%)	1.5328	7.1319	1.6497	3.1373	8.0360	4.5344	5.2691	4.3162	6.8650	3.3439	8.0846	14.3207	12.0246	11.4091	6.54678
	Ave. time (ms)	843	7000	13,611	11,240	11,687	7937	15,062	12,109	13,298	11,562	13,843	17,437	14,657	19,153	12,103
INS	Best Err. (%)	0.0663	0.2162	0.1862	0.9498	2.5029	1.5906	2.8412	2.1800	3.1463	2.3002	2.1065	9.1262	8.4822	8.0921	3.12761
	Ave. Err. (%)	1.2450	1.6028	1.0515	1.5028	4.6908	2.9788	5.0280	3.3973	4.6438	3.5785	4.3723	11.7217	10.7559	9.4238	4.71378
	Ave. time (ms)	843	7003	13,609	11,240	11,687	7937	15,064	12,109	13,297	11,562	13,843	17,437	14,656	19,141	12,102
SIM	Best Err. (%)	**0.0000**	0.3477	0.2255	0.9957	2.4799	1.7535	1.7650	1.7617	2.4482	1.8469	1.6088	6.4203	6.5312	7.7534	2.56699
	Ave. Err. (%)	1.0594	1.9857	0.4990	1.6023	3.5545	2.7499	3.5213	2.7237	4.0319	2.3377	3.7933	8.8180	8.6754	8.8103	3.86876
	Ave. time (ms)	746	7000	13,609	11,240	11,687	7937	15,062	12,116	13,297	11,562	13,843	17,437	14,657	19,141	12,095
SCM	Best Err. (%)	**0.0000**	1.8043	0.2153	1.5625	3.3754	1.7766	5.6823	2.8581	2.8160	3.1956	2.5818	10.7697	6.7834	7.7967	3.65840
	Ave. Err. (%)	1.5845	3.7031	0.7428	2.3744	6.7099	2.7410	7.0728	4.6318	4.2863	3.5539	5.0996	13.0749	9.8846	10.6779	5.43838
	Ave. time (ms)	743	7000	13,609	11,240	11,687	7940	15,062	12,109	13,297	11,565	13,843	17,437	14,656	19,141	12,095
GSM	Best Err. (%)	**0.0000**	3.0448	0.2255	1.6238	3.9878	1.8417	5.4671	2.9341	3.3710	3.0622	3.9431	10.2135	9.1342	5.9868	3.91682
	Ave. Err. (%)	1.0528	5.4304	**0.3306**	3.0499	5.5499	3.3446	7.0814	4.3790	4.9006	4.1454	6.1249	13.3050	11.0807	7.8599	5.54536
	Ave. time (ms)	682	7000	13,609	11,240	11,687	7937	15,062	12,109	13,296	11,564	13,843	17,437	14,656	19,141	12,090
GSTM	Best Err. (%)	**0.0000**	**0.0000**	**0.0000**	**0.4596**	**0.9644**	**0.7695**	**0.6027**	**0.3866**	**0.8683**	**0.2527**	**0.7242**	**1.2326**	**0.9827**	**2.0501**	**0.66380**
	Ave. Err. (%)	**0.0000**	**1.1836**	1.0809	**0.6357**	**1.7616**	**1.6202**	**1.8425**	**1.2193**	**1.5432**	**0.4994**	**1.5287**	**2.9169**	**3.3099**	**2.7758**	**1.56554**
	Ave. time (ms)	836	6987	13,598	11,240	11,684	7937	15,050	12,096	13,292	11,559	13,843	17,424	14,643	19,132	12,094

3. Greedy Sub Tour Mutation operator

The purpose of mutation in GAs is to allow the algorithm to avoid local solutions and to make population have new previously unprecedented examples. Exchange Mutation (EXC) (Banzhaf, 1990), Displacement Mutation (DISP) (Michalewicz, 1992), Inversion Mutation (INV) (Fogel, 1993a, 1993b), Insertion Mutation (INS) (Fogel, 1988; Michalewicz, 1992), Simple Inversion Mutation (SIM) (Grefenstette, Gopal, Rosmaita, & Gucht, 1985; Holland, 1975), Scramble Mutation (SCM) (Syswerda, 1991) and other mutation operators provide previously unprecedented examples to the population by making randomly distortion on individuals. So that during the optimization process the probability of encountering local solutions is decreased. But, these operators have no efforts to develop the tour while they are implementing the distortion operation. The operators which choose and apply the developments that provide maximum gain just for that instant are called as greedy methods. Greedy algorithm such as Greedy Swap Mutation (GSM) (Louis & Tang, 1999) also selects better result and therefore comes up closer to the solution quickly. However, when greedy methods reach a local solution it is very difficult to jump up any other unprecedented solutions. Therefore greedy methods produce local solutions that are so far from optimal.

GSTM operator that is developed for TSP solution by GA uses classical and greedy techniques together by the aid of different parameters. Parametrical structure of GSTM prevents a stuck of the local solutions like other greedy mutation operators and provides faster access to solutions. The GSTM operator reaches local solutions faster in a completely far from randomness way by greedy search methods and making random distortion on these solutions like classical mutation operators and produces early unprecedented examples and then examines them again by greedy way.

Six parameters in the GSTM operator determine working format and direction of mutation operator. The GSTM algorithm is described in Fig. 2.

The parameters L_{MIN} and L_{MAX} are used to define the size of the sub-tour that is necessary to be selected during mutation operation. L_{MIN} and L_{MAX} present smallest and biggest size that can be selected for the sub-tour, respectively. The sub-tour (T^*) with $R_1 - R_2$ interval that will be selected on the individual under mutation operation has to be between L_{MIN} and L_{MAX} ($L_{MIN} \leqslant |R_1 - R_2| \leqslant L_{MAX}$). Maximum sub-tour distance that can be selected with L_{MAX} depends on the number of the cities (n) in the TSP.

Reconnection probability (P_{RC}) is the working probability of method that provides removing selected T^* sub-tour from T tour and reconnecting it to the $T^{\#}$ tour such a way that minimum extension should occur (Fig. 3).

If the reconnection operation that is a greedy operation is undone, then either limited inversion of sub-tour with using search space neighbor lists or random distortion for the population to have the unprecedented examples will be done according to the correction and distortion probability (P_{CP}). During the process of rotation of sub-tour that uses neighborhood list, firstly, one each neighbors (NL_{R1} and NL_{R2}) are chosen from neighbor list that belongs to both of the sub-tour points (R_1 and R_2) and whose size (NL_{MAX}) has been determined previously. Then, inversion (G_{R1} and G_{R2}) are applied to these neighbors such a way that they will be neighbors to sub-tour point and maximum gain among the different tours occurred (Fig. 4). The neighbor lists are recorded separately for each city c_i and it contains NL_{MAX} number of neighbor cities that are nearest to the city c_i.

That is necessary to make a random distortion on individual that is subject of mutation as a precaution to the probability of being stuck up to the local solutions. Depending on P_{CP} and probability of linearity (P_L) the distortion which will be applied to the individual is made. In the T tour, the sub-tour (T^*) between the points R_1 and R_2 is distorted in itself. P_L defines regularity of distortion operation that will be applied. As P_L comes close to 1, the probability of mixing of each city randomly will increase like Scramble mutation operator whereas when it comes close to 0, probability of making a normal inversion operation increases like Simple Inversion Mutation operator (Fig. 5).

4. Experimental results

For experiments a computer with Intel Pentium IV 2.8 GHz processor was used and application software was developed in Borland Delphi 7. To test developed operator performance, 14 different TSP were taken from the TSPLIB (TSPLIB, 2009) and 10 trials were examined for each problem.

During GA applications size of the population is selected as $P = 40$ and initial population is produced by Nearest Neighbor tour construction heuristic. As selection method Tournament Selection is used ($k = 5$). Crossover method and probability (P_C) ratio are chosen as DPX (Freisleben & Merz, 1996) and 0.8, respectively.

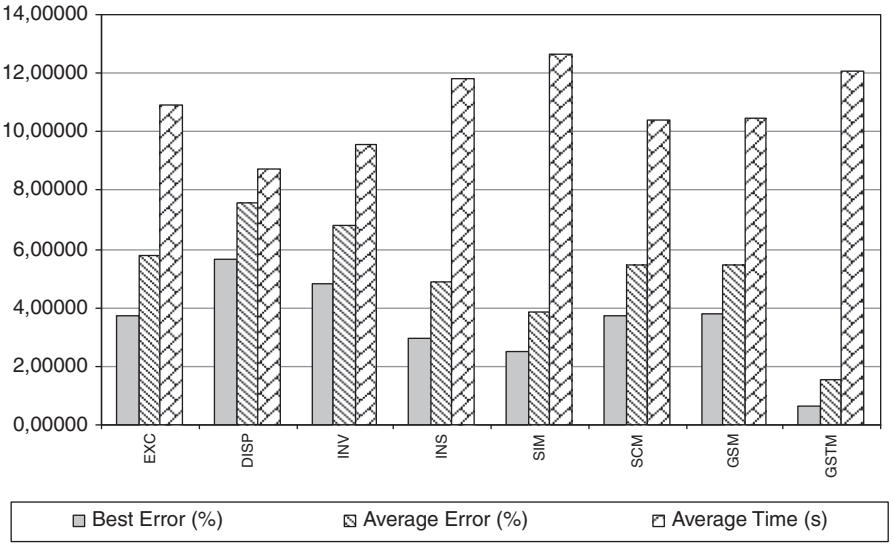


Fig. 6. Performance of mutation methods.

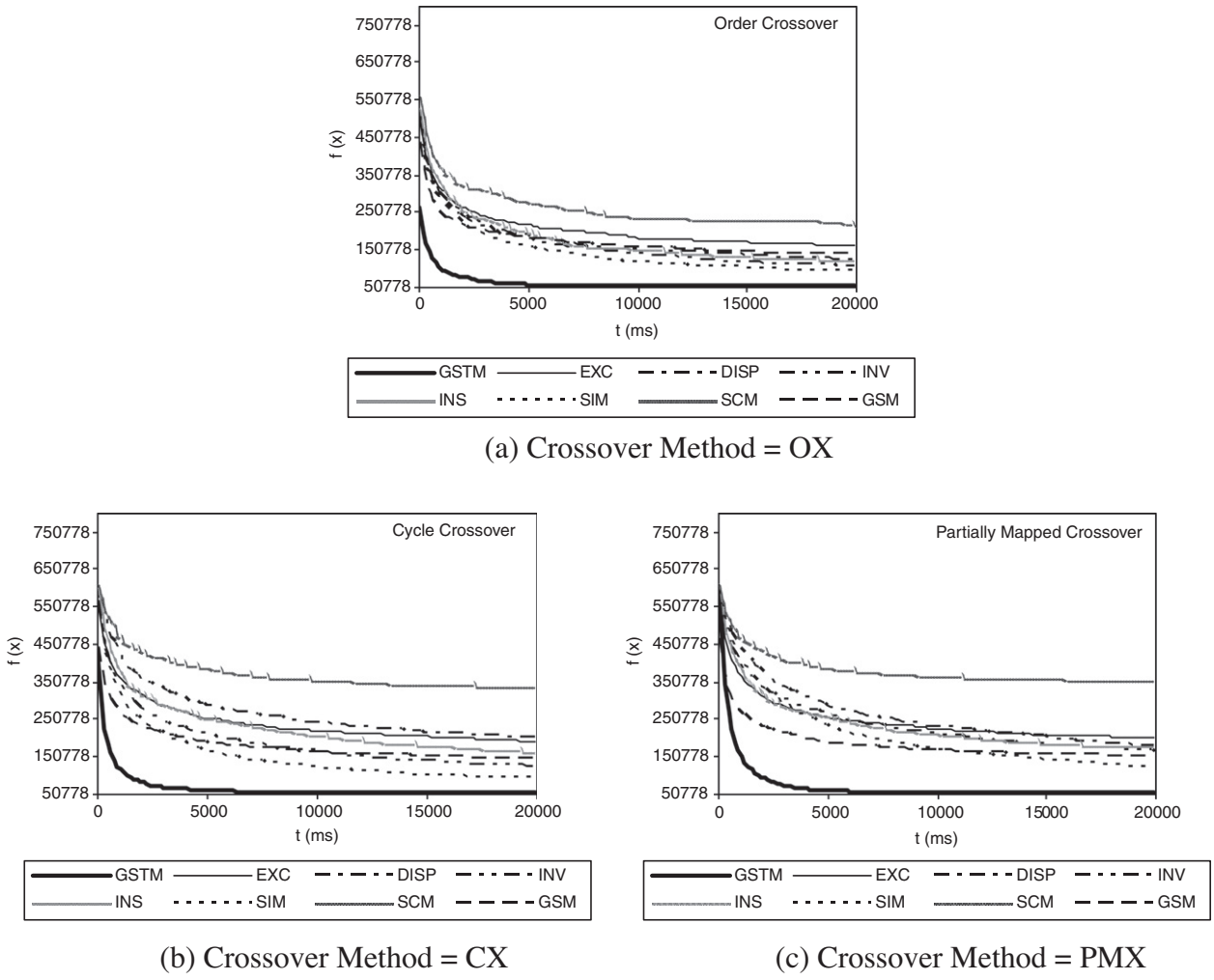


Fig. 7. Experiment with different methods of mutation operators.

Mutation method and probability (P_M) ratio are chosen as proposed GSTM and 0.2, respectively.

As the results of the investigations, followings are obtained ideal parameters values of GSTM operator used with GA;

$\text{GSTM}_{GA} : P_{RC} = 0.5, \quad P_{CP} = 0.8, \quad P_L = 0.2, \quad L_{MIN} = 2,$

$L_{MAX} = Int(\sqrt{n}), \quad NL_{MAX} = 5.$

The effects of eight different mutations operator included proposed GSTM with GA were examined in 14 different TSP solutions. Analysis that has best termination (stopping) condition by selecting 20,000 iterations where the best individual does not change showed that the produced errors of GSTM are: in the best situation between 0% and 2.05010% and in case of average situation between 0% and 3.30986%. So, we can say that the GSTM operator has given better results for these values regarding to the other operators (Table 1). In the other analysis, to see the difference running times, termination condition is chosen as equal running time for all operators (Table 2). Using the average values of the results for 14 TSP in the Tables 1 and 2 we can obtain a general mutation performance (Fig. 6).

Besides, we can clearly see from Fig. 7 and analyses performed on the PCB442 problem that GSTM produces lower error rates and faster results by using different crossover operators like Partially Mapped Crossover (PMX) (Goldberg & Lingle, 1985), Order Crossover (OX) (Davis, 1985) and Cycle Crossover (CX) (Oliver, Smith, & Holland, 1987).

5. Conclusion

Obtained results in this study showed that the developed GSTM operator is much more successful with respect to the error and speed regarding to EXC, DISP, INV, INS, SIM, SCM and GSM mutation operators. During the exploration process where termination condition is selected as not to change the best individual in population during 20,000 iterations, it is occurred that GSTM operator produced much better solutions in best error value (0.66380%) and in average error value (1.56554%) regarding to the all other standard mutation operators (Table 1). GSTM decreased best error value of the second best mutation method (SIM) with the ratio of approximately 73.24%, average error value 59.42% and running time approximately 8.24%. Furthermore, GSTM decreased best error value of the DISP operator that produces highest error value with the ratio of approximately 88.32% and average error value approximately 79.51% (see Table 1).

In the analysis where equal time is given for all mutation methods it is also seen that GSTM method has best solutions (Table 2). GSTM decreased best error value of SIM operator that produces very close error values to GSTM with the ratio of approximately 74.14% and average error value 59.53%. Also, GSTM decreased best error value of the DISP operator that produces highest error value with the ratio of approximately 88.29% and average error value approximately 79.15% (see Table 2).

So, the GSTM operator reduces the best error values with the ratio of between 74.14% and 88.29%, average error values between 59.93% and 79.15% in comparison with the simple GA that used with other mutation operators.

As result one can see that GSTM increases the performance of GA in TSP solutions regarding to the other mutation operators (Fig. 6) remarkably. Moreover, it was seen that it produces better results when it is used with other simple crossover operators that do not produce good results with other mutation operators. It may be useful to install an adaptive component to the GSTM for defining of parameters during optimization.

Acknowledgement

This study is supported as research project by the Unit of Scientific Research Projects of Selcuk University in Turkey (Project No. 07201054/2008).

References

Albayrak Murat (2008). Determination of route by means of Genetic Algorithms for printed circuit board driller machines. *Master dissertation* (p. 180). Selcuk University.

Banzhaf, W. (1990). The 'molecular' traveling salesman. *Biological Cybernetics, 64*, 7–14.

Davis, L. (1985). Applying adaptive algorithms to epispastics domains. In *Proceedings of the int. joint conf. on artificial intelligence (IJCAI'85)* (pp. 162–164). Los Angeles, CA.

Dorigo, M., & Gambardella, L. M. (1997). Ant colonies for the traveling salesman problem. *BioSystems, 43*, 73–81.

Fogel, D. B. (1988). An evolutionary approach to the traveling salesman problems. *Biological Cybernetics, 60*, 139–144.

Fogel, D. B. (1993a). Empirical estimation of the computation required to discover approximate solutions to the traveling salesman problem using evolutionary programming. In *Proceedings of 2nd annual conference on evolutionary programming* (pp. 56–61).

Fogel, D. B. (1993b). Applying evolutionary programming to selected traveling salesman problems. *Cybernetics and Systems: An International Journal, 24*, 27–36.

Freisleben, B., & Merz, P. (1996). A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *Proceedings of the IEEE international conference on evolutionary computation* (pp. 616–621). IEEE Press.

Genetic Algorithms (2009). Wikipedia. http://en.wikipedia.org/wiki/Genetic_Algorithms; last access.

Goldberg, D. E., & Lingle, R. (1985). Alleles, loci and the traveling salesman problem. In *Proceedings of the first int. conf. on genetic algorithms (ICGA'85)* (pp. 154–159). Pittsburgh, PA: Carnegie-Mellon University.

Grefenstette, J., Gopal, R., Rosmaita, B., Gucht, D. V. (1985). Genetic Algorithms for the TSP. In *Proceedings of the first international conference on genetic algorithms and their applications* (pp. 160–65).

Helsgaun, K. (2000). An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research, 126*(1), 106–130.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

Jayalakshmi, G. A., Sathiamoorthy, S., & Rajaram, R. (2001). A hybrid genetic algorithm – a new approach to solve traveling salesman problem. *International Journal of Computational Engineering Science, 2*(2), 339–355.

Louis, S. J., Tang, R. (1999). Interactive genetic algorithms for the traveling salesman problem. In *Proceedings of the 1999 genetic and evolutionary computing conference (GECCO 1999)* (pp. 385–392).

Michalewicz, Z. (1992). *Genetic algorithms + data structures = evolution programs*. Berlin: Springer.

Misevicius, A. (2004). Using iterated tabu search for the traveling salesman problem. *Information Technology and Control, 3*(32), 29–40.

Oliver, I. M., Smith, D. J., & Holland, J. R. C. (1987). A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the second int. conf. on genetic algorithms (ICGA'87)* (pp. 224–230). Cambridge, MA: Massachusetts Institute of Technology.

Schleuter, M. G. (1997). Asparagos96 and the traveling salesman problem. In *Proceedings of the 1997 IEEE international conference on evolutionary computation* (pp. 171–174). IEEE Press.

Seo, D., Moon, B. (2002). Voronoi quantized crossover for traveling salesman problem. In *Proceedings of the genetic and evolutionary computation conference* (pp. 544–552).

Stützle, T., Hoos, H. (1997). The MAX–MIN ant system and local search for the traveling salesman problem. In *Proceedings of the IEEE international conference on evolutionary computation* (pp. 308–313). Indianapolis, Indiana, USA.

Syswerda, G. (1991). Schedule optimization using genetic algorithms. In L. Davis (Ed.), *Handbook of genetic algorithms* (pp. 332–349). Van Nostrand Reinhold.

TSPLIB (2009). http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/; last access: April 2009.