A Brief Review of Memetic Algorithms for Solving Euclidean 2D Traveling Salesrep Problem

Ender Ozcan and Murat Erenturk

Department of Computer Engineering
Yeditepe University
Istanbul, 34755, Turkey

Abstract. Traveling Salesrep Problem (TSP) is one of the classical combinatorial optimization problems. There is a variety of approaches for finding near optimal solutions to problem instances. In this paper, a survey of Memetic Algorithms (MAs) is provided. Some experiments are performed on small TSP instances using traditional MAs using various operators, including a modified two-point crossover. The best combination of these operators and parameters settings are used to solve TSP on Turkish cities.

1 Introduction

Traveling Salesrep Problem (TSP) is a typical combinatorial optimization problem, studied by numerous researchers. A salesrep is required to visit N cities exactly for once, completing a tour by arriving at any city that is also the start and travelling the minimum distance. More formally, given N cities, TSP requires a search for a permutation $\pi : \{0,\ldots,N-1\} \rightarrow \{0,\ldots,N-1\}$, using a cost matrix C=[c_{ij}], where c_{ij} denotes the cost (assumed to be known by the salesrep) of the travel from city i to j, that minimizes the *path length*

$$f(\pi, C) = \sum_{i=0}^{N-1} c_{\pi(i),\pi((i+1) \bmod N)}, \qquad \text{Equation 1.}$$

where $\pi(i)$ denotes the city at i^{th} location in the tour.

Different classes of TSP can be identified by the properties of the cost matrix. In *symmetric* TSP c_{ij}=c_{ji}, $\forall i, j$, otherwise this set of problems are referred as *asymmetric* TSP. If the cities lie in a metric space, satisfying the triangle inequality, the problem is referred as metric TSP. Assuming that each city in a tour is marked by its position (x_i,y_i) in the plane, and the cost matrix C contains the Euclidean distances between the i^{th} and j^{th} city:

$$c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \qquad \text{Equation 2.}$$

Then the problem is both symmetric and metric.

The search space of a Euclidean TSP is giant containing N! permutations and identified by Garey *et. al.* in [8] to be NP-hard. There are many exact and

approximation algorithms for solving TSP. Having a variety of application areas, such as, vehicle routing, robot control, crystallography, computer wiring, scheduling, etc. and being a combinatorial optimization problem, TSP attracts the attention of Genetic Algorithm (GA) community.

2 Memetic Algorithms and Previous Work

Genetic Algorithms (GAs) were introduced by J. Holland in [11], and have been used to solve many difficult problems (Goldberg [9]). A *population* of *individuals*, representing candidate solutions, is evolved from an initial *generation* towards a final generation. An *evaluation function* is used to measure how fit an individual is. At each evolutionary step, a pool of individuals for mating is selected; giving higher chance for fitter individuals, and then a *crossover* operator is applied, producing *offspring*, on which *mutation* is applied. S*election pressure* lets fitter individuals to survive after the *replacement*. The evolution continues until some *termination criteria* are satisfied. Memetic Algorithms (MAs), introduced by Moscato *et*. *al*. in [20] and formalized by Raddcliff *et*. *al*. in [27], extend GAs by applying a *local search* on individuals after mutation. Usefulness of *hill climbing* in search algorithms for problem solving is emphasized by many researchers, such as, Alkan *et*. *al*. in [2], Ozcan *et*. *al*. in [25], [26].

Memetic Algorithms (MAs) are also used for solving TSP by combining local search techniques and GAs. Aarts *et*. *al*. in [1] provides some test results of MAs for the TSP, using 2-opt (Lin [17]) and variable depth neighbourhoods (Lin *et*. *al*. [18]) as local search techniques. Larranaga *et*. *al*. in [9] makes a review of representations and operators used in Genetic Algorithms for solving TSP. Table 1 is generated based on the findings in [9], showing a subset of genetic operators used in solving TSP.

There are several schemes to represent a candidate solution to a TSP instance: *binary representation*, *path representation*, *adjacency representation*, *ordinal representation*, and *matrix representation*. Ucoluk introduces a new scheme in [32], based on inversion sequences, unnecessitating a special crossover or mutation operator for solving TSPs.

Table 1. Some genetic operators used for solving TSPs.

Operator Name	Year	Authors
Alternating Position Crossover (AP)	(1999)	Larranaga, Kuijpers, Poza and Murga [16]
Cycle Crossover (CX)	(1987)	Oliver, Smith and Holland [24]
Distance Preserving Crossover (DPX)	(1996)	Freisbein and Merz [7]
Edge Assembly Crossover (EAX)	(1997)	Nagata and Kobayashi [23]
Edge Recombination Crossover (ER)	(1989)	Whitley, Timothy and Fuquay [31]
Heuristic Crossover (HEU)	(1987)	Grefenstette [14]
Inver-over Operator (IOO)	(1998)	Tao and Michalewicz [30]
Maximal Preservative Crossover (MPX)	(1988)	Mühlenbein, Schleuter and Krämer [21]
Position Based Crossover (POS)	(1991)	Syswerda [29]
Order Crossover (OX1)	(1985)	Davis [4]
Order Based Crossover (OX2)	(1991)	Syswerda [29]
Partially mapped Crossover (PMX)	(1985)	Goldberg and Lingle [10]
Voting Recombination Crossover (VR)	(1989)	Mühlenbein [22]

Displacement Mutation (DM)	(1992)	Michalewicz [19]
Exchange Mutation (EM)	(1990)	Banzhaf [3]
Insertion Mutation (ISM)	(1988)	Fogel [5]
Inversion Mutation (IVM)	(1990)	Fogel [6]
Scramble Mutation (SM)	(1991)	Syswerda [29]
Simple Inversion Mutation (SIM)	(1975)	Holland [11]

There are some new operators, having better performances that are not mentioned in [9]. Nagata *et. al.* describes a powerful crossover in which local search is performed during the application of this genetic operator in [23]. Similarly, Seo *et. al.* presents another intelligent crossover operator in [28], called Voronoi Quantized Crossover (VQX), based on *genic distances*. Tao *et. al.* introduces a unary operator based on simple inversion, yielding fast results utilizing a nonstandard evolutionary algorithm in [30]. Also, Freisbein *et. al.* uses a new crossover operator in a nonstandard GA, producing an offspring by preserving the Hamming distance between tours. Hence, the number of newly introduced edges is minimized. Jung *et. al.* uses the Natural Crossover (NX) on images, utilizing topological information for solving 2D Euclidean TSP instances in [12].

Krasnogor *et. al.* applies local search after the individuals are selected for the next generation and during initialization in [13]. The main feature of the MAs is the utilization of different schemes for accepting and rejecting the improvements during the local search. As a hill climbing step 2-swap is used in the experiments.

In this paper, we explore the effect of crossover, mutation and local search in Genetic Algorithms. The best combination of operators is used to solve a TSP, in which sales representative travels between Turkish cities.

3 Memetic Algorithms for TSP

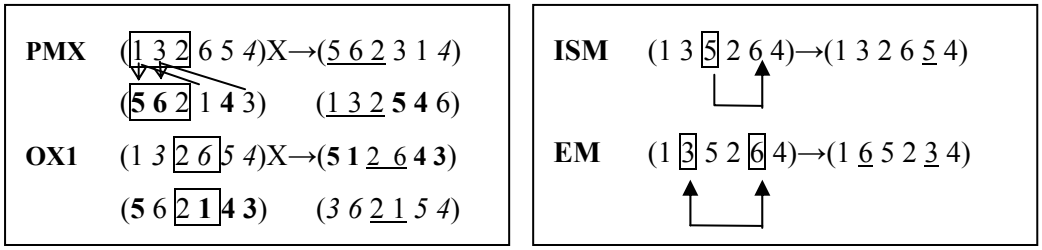
An individual uses *path representation* in our MA. For example, assuming a 6-city TSP, (1 3 5 2 6 4) represents a tour starting from city 1, visiting 3, 5, 2, 6, 4 in that order and returning back to 1. Initial population is created randomly. Partially Mapped Crossover (PMX), Order Crossover (OX1) and two-point crossover (2PTX) are implemented. After 2PTX is applied, a patch-up operator randomly assigns unvisited cities for the regions requiring a repair as follows:

Parent1: 1 3 5 7 [4 6] 2 Offspring1: x x 5 7 [1 3] 2→ Offspring1: 6 4 5 7 [1 3] 2

Parent2: 2 7 5 4 [1 3] 6 Offspring2: 2 7 5 x [4 6] x→ Offspring1: 2 7 5 1 [4 6] 3

Note that 2PTX and the designed patch up operator might yield the same result as OX1 in some cases. This operator works like a combined and modified version of order crossover and scramble mutation. Additional to these operators, insertion mutation (ISM) and swap mutation (EM) operators are realized. Figure 1 demonstrates how these genetic operators work. Three different methods for selecting mates are implemented: Ranking (RANK), tournament (TOUR) and random (RAND).

Furthermore two types of MAs are implemented: Steady State Memetic Algorithm (SSMA) and Trans-generational Memetic Algorithm (TGMA). SSMA utilizes an elitist survival strategy, replacing worst pair of individuals with the best pair among the offspring and themselves. TGMA saves a small best portion of a population into the next generation and fills out the rest with newly produced offspring.



(a) **(b)**

Figure 1. Example operations of **(a)** PMX and OX1, and **(b)** ISM and EM.

As a hill climbing approach, a hill climbing step is applied as long as, maximum number of iterations is not exceeded and the individual is improved. Each hill climbing step is chosen to be the mutation operator. Note that the best known local search algorithm is Lin-Kernighan (LK) algorithm (Lin *et. al*. in [18]). Evolution is terminated either a maximum number of generations is exceeded or an *expected fitness* is achieved. Fitness function evaluates the total path length as in Equation 1. Population size is a multiple of the number of cities in the TSP instances.

4 Experiments

Initial experiments are performed to achieve the best combination of operators for MA. Then the final MA is used to solve using Turkish cities. All the experiments are repeated for 100 times.

4.1 Experimental Data

As an experimental data, five syntactic data, labeled as C20, C30, C40, S21, F32, F41, and a real data, labeled as T81 are used, where each integer valued suffix indicate the number of cities in the TSP instance. In C-class data, all the cities are are placed on a circle as shown in Figure 2.(a), equidistantly. Similarly, S21 is a square with a side length of 15000 units. F-class data are fractals as revealed in Figure 2.(b) and (c). Figure 2.(d) shows the locations of 81 cities in Turkey based on their longitude and latitudes. Optimum path lengths that are used as expected fitness values in MAs for each TSP instance is presented in Table 2.

4.2 Experimental Results

All possible combinations of operators are tested. Performance of crossover operators; OX1, PMX and 2PTX in SSMA and TGMA is summarized in Table 3, in terms of average fitness per generation in a run averaged over 100 runs. On average, results demonstrate that OX1 performs the best. Interestingly, 2PTX with patch up performs better than PMX in most of the cases. If the mutation operators; ISM and EM are compared, based on the results mentioned in Table 4, EM turns out to be better than ISM, on average. Test results of selection methods for recombination; tournament, random and ranking in SSMA and TGMA is provided in Table 5, in terms of average fitness per generation in a run averaged over 100 runs. On average, results show that TOUR is the best choice as a selection method. As expected RAND is the worst choice among them.

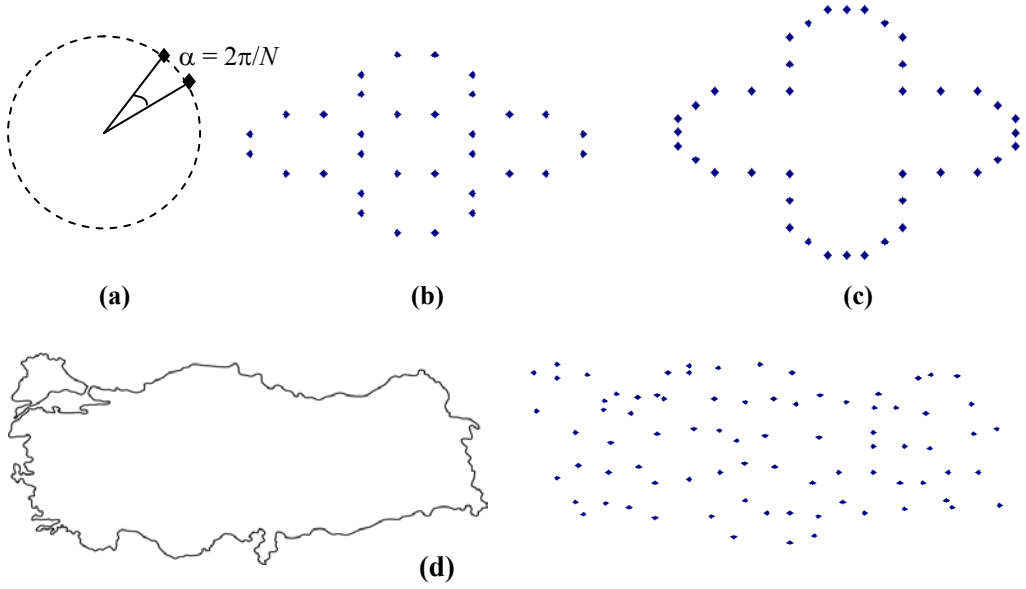


Figure 2. Experimental data: **(a)** Cities on a circle (C20, C30, C40), **(b)** F32, **(c)** F41, **(d)** Cities in Turkey (T81).

Table 2. Expected fitness values for TSP instances used in the experiments.

Data Label (D.L.)	Expected Fitness	D.L.	Expected Fitness
C20	62.575	S21	60.000
C30	62.716	F32	84.180
C40	62.768	F41	68.168

All the tables illustrate the success of Memetic Algorithms as compared to Genetic Algorithms in solving TSP. Even using a simple hill climbing operator as mentioned in Section 3 yields better results than the traditional GA. Best combination of operators is used to solve TSP for Turkish cities and the following path is obtained:

[HAKKARİ, ŞIRNAK, SİİRT, BİTLİS, MUŞ, BİNGÖL, ERZİNCAN, TUNCELİ, ELAZIĞ, DİYARBAKIR, BATMAN, MARDİN, ŞANLIURFA, ADIYAMAN, MALATYA, KAHRAMANMARAŞ, GAZİANTEP, KİLİS, HATAY (Antakya), OSMANİYE, ADANA, KAYSERİ, YOZGAT, NEVŞEHİR, NİĞDE, İÇEL(Mersin), KARAMAN, ANTALYA, BURDUR, AFYON, ISPARTA, KONYA, AKSARAY, KIRŞEHİR, KIRIKKALE, KARABÜK, BARTIN, ZONGULDAK, BOLU, DÜZCE, SAKARYA(Adapazarı), BİLECİK, KÜTAHYA, UŞAK, DENİZLİ, MUĞLA, AYDIN, İZMİR, MANİSA, BALIKESİR, ÇANAKKALE, EDİRNE, KIRKLARELİ, TEKİRDAĞ, İSTANBUL, BURSA, YALOVA, KOCAELİ(İzmit), ESKİŞEHİR, ANKARA, ÇANKIRI, KASTAMONU, ÇORUM, SİNOP, AMASYA, SAMSUN, TOKAT, SİVAS, ORDU, GİRESUN, GÜMÜŞHANE, TRABZON, BAYBURT, ERZURUM, RİZE, ARTVİN, ARDAHAN, KARS, AĞRI, IĞDIR, VAN]

Table 3. Experimental results utilizing different GA types and crossover operators, where **α** indicates average fitness per generation at each run, averaged over 100 runs, and **β** indicates the best fitness values achieved during these runs.

GA TYPE	D.L.	OX1		PMX		2PTX	
		α	**β**	**α**	**β**	**α**	**β**
	C20	**108.028**	**62.575**	146.258	77.496	130.375	62.998
	C30	118.658	**63.048**	198.126	62.716	**118.797**	63.395
SSGA	C40	**131.864**	**62.768**	296.656	**62.768**	133.651	63.499
	S21	**93.915**	**60.000**	121.431	**60.000**	107.623	**60.000**
	F32	**111.713**	101.277	193.130	91.094	112.130	**100.088**
	F41	**122.177**	92.025	288.901	**77.609**	128.192	92.025
	C20	**107.493**	**62.575**	145.761	68.596	137.425	**62.575**
	C30	**116.633**	**62.716**	232.692	**62.716**	126.822	**62.716**
SSGA	C40	128.117	63.117	326.627	**62.768**	**127.315**	**62.768**
HC	S21	**93.626**	**60.000**	130.871	**60.000**	118.905	**60.000**
	F32	**108.048**	97.347	191.778	**89.288**	108.371	98.640
	F41	115.860	89.079	278.021	**68.168**	**115.703**	**68.168**
	C20	**155.053**	**62.575**	200.678	**62.575**	155.642	**62.575**
	C30	199.731	62.716	276.245	**60.000**	**199.010**	63.142
TGGA	C40	**242.713**	**62.768**	366.881	**62.768**	325.438	**62.768**
	S21	**139.622**	**60.000**	182.676	**60.000**	158.829	**60.000**
	F32	**172.393**	92.945	231.142	**84.180**	173.531	94.675
	F41	**209.223**	**68.168**	313.109	**68.168**	271.056	**68.168**
	C20	134.497	**62.575**	181.913	**62.575**	**134.348**	**62.575**
	C30	165.795	**62.716**	248.582	**62.716**	**164.665**	**62.716**
TGGA	C40	**197.829**	**62.768**	317.268	**62.768**	373.212	**62.768**
HC	S21	**120.276**	**60.000**	165.421	**60.000**	139.039	**60.000**
	F32	**146.317**	**84.180**	205.101	101.886	147.306	88.151
	F41	**158.461**	**68.168**	268.375	**68.168**	281.333	**68.168**

Table 4. Experimental results utilizing different GA types and mutation operators, where **α** indicates average fitness per generation at each run, averaged over 100 runs, and **β** indicates the best fitness values achieved during these runs.

GA TYPE	D.L.	ISM		EM	
		α	**β**	**α**	**β**
SSGA	C20	145.655	**62.575**	**110.786**	71.431
	C30	147.590	**62.716**	**142.798**	**62.716**
	C40	207.804	**62.768**	**157.104**	121.454
	S21	116.598	**60.000**	**98.714**	**60.000**
	F32	159.191	**91.094**	**118.792**	108.694
	F41	207.856	**77.609**	**151.658**	120.573
SSGA HC	C20	149.799	**62.575**	**110.654**	**62.575**
	C30	186.314	**62.716**	**131.118**	88.975
	C40	243.077	**62.768**	**144.962**	114.914
	S21	129.523	**60.000**	**99.411**	**60.000**
	F32	157.291	**89.288**	**114.841**	106.003
	F41	205.571	**68.168**	**134.152**	92.426
TGGA	C20	173.683	**62.575**	**167.233**	**62.575**
	C30	236.148	62.716	**213.842**	**60.000**
	C40	344.740	**62.768**	**278.614**	**62.768**
	S21	169.553	**60.000**	**151.198**	**60.000**
	F32	201.821	92.945	**182.890**	**84.180**
	F41	287.101	**68.168**	**241.825**	**68.168**
TGGA HC	C20	153.128	**62.575**	**147.378**	**62.575**
	C30	198.448	**62.716**	**187.580**	**62.716**
	C40	309.523	**62.768**	**282.683**	**62.768**
	S21	149.920	**60.000**	**133.237**	**60.000**
	F32	172.591	**84.180**	**159.893**	86.734
	F41	**239.901**	**68.168**	241.260	**68.168**

Table 5. Experimental results utilizing different GA types and selection methods for recombination, where **α** indicates average fitness per generation at each run, averaged over 100 runs, and **β** indicates the best fitness values achieved during these runs.

GA TYPE	D.L.	TOURNAMENT		RANDOM		RANKING	
		α	**β**	**α**	**β**	**α**	**β**
	C20	**126.436**	**62.575**	126.944	**62.575**	131.282	**62.575**

	C30	140.705	**62.716**	173.790	**62.716**	**121.087**	**62.716**
SSGA	C40	177.456	**64.008**	228.010	64.034	**128.714**	62.768
	S21	**101.145**	**60.000**	114.465	**60.000**	107.359	**60.000**
	F32	**128.573**	91.623	150.232	101.646	138.168	**91.094**
	F41	**160.240**	**77.609**	201.873	92.138	177.157	92.025
	C20	**113.707**	**62.575**	143.931	**62.575**	133.041	**62.575**
	C30	**141.000**	**62.716**	182.180	63.048	152.969	63.525
SSGA	C40	**177.561**	**62.768**	217.941	63.117	186.557	**62.768**
HC	S21	106.584	**60.000**	130.238	**60.000**	**106.579**	**60.000**
	F32	127.228	**89.288**	146.701	97.347	**134.269**	91.094
	F41	157.085	80.433	186.761	**68.168**	**165.738**	**68.168**
	C20	**134.429**	**62.575**	224.846	71.359	152.098	**62.575**
	C30	**153.537**	**60.000**	330.044	90.165	191.405	63.852
TGGA	C40	**228.951**	**62.768**	476.435	65.886	229.646	**62.768**
	S21	**120.017**	**60.000**	201.758	**60.000**	159.352	**60.000**
	F32	**141.048**	**84.180**	268.649	111.198	167.369	96.122
	F41	**162.891**	**68.168**	379.565	78.357	250.932	**68.168**
	C20	**120.202**	**62.575**	193.156	**62.575**	137.401	**62.575**
	C30	**130.820**	**62.716**	278.075	**62.716**	170.147	**62.716**
TGGA	C40	**151.887**	**62.768**	428.456	**62.768**	307.965	**62.768**
HC	S21	**106.970**	**60.000**	172.790	**60.000**	144.976	**60.000**
	F32	**117.335**	**84.180**	232.659	**84.180**	148.732	86.734
	F41	**121.668**	**68.168**	352.459	**68.168**	266.375	**68.168**

5 Conclusions

Considering path representation, several different and well known operators are tested utilizing Genetic Algorithms and Memetic Algorithms to solve some small instances of Travelling Salesrep Problem in 2D, combining with hill climbing. Emprical results yield the success of following operators from the best to the worst: OX1, 2PTX and PMX as crossover, and EM and ISM as mutation and hill climbing method. Note that 2PTX uses a patch-up algorithm, producing a modified crossover combining OX1 and scramble mutation that has not been tested before. Hence, the best combination of operators is OX1 and EM for both TGGA and SSGA. Furthermore, results show that hill climbing improves the solution in any GA type. Using a transgenerational memetic

algorithm with OX1 and EM, TSP is solved for Turkish cities. To our knowledge, this is the first time; the optimal path is obtained for this instance.

References

1. E. H. L. Aarts and M. G. A. Verhoeven, Genetic local search for the traveling salesman problem, *Handbook of Evolutionary Computation*, pp.G9.5:1-7, IOP publishing Ltd and Oxford University Press. (1997)
2. A. Alkan, E. Ozcan, Memetic Algorithms for Timetabling, *IEEE Congress on Evolutionary Computation*, pp. 1796-1802. (2003)
3. W. Banzhaf, The "molecular" traveling salesman, *Biological Cybernetics*, vol. 64, pp. 7–14. (1990)
4. L. Davis, Applying adaptive algorithms to epistatic domains, *Proceedings of the International Joint Conference on Artificial Intelligence*, vol. 1, pp. 161– 163. (1985)
5. D. B. Fogel, An evolutionary approach to the travelling salesman problem, *Biological Cybernetics*, vol. 60, no. 2, pp. 139–144. (1988)
6. D. Fogel, A parallel processing approach to a multiple travelling salesman problem using evolutionary programming, *Proceedings of the Fourth annual Symposium on Parallel Processing*, (Fullerton, California), pp. 318–326. (1990)
7. B. Freisleben, and P. Merz, New Genetic Local Search Operators for the Traveling Salesman Problem, *Proc. of 4th Conf. on Parallel Problem Solving from Nature - PPSN IV*, vol. 1141, pp. 890-900, Springer. (1996)
8. M. R. Garey, R.L. Graham, and D. S. Johnson. Some NP-complete geometric problems*, In 8th Annual ACM Symposium on Theory of*, pp 10-22. (1976)
9. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading (MA). (1989)
10. D. E. Goldberg, and R. Lingle Jr., Alleles, loci, and the traveling salesman problem, in *Proceedings of the First International Conference on Genetic Algorithms and Their Applications* (J. J. Grefenstette, ed.), Lawrence Erlbaum Associates, Publishers. (1985)
11. J. H. Holland, *Adaptation in Natural and Artificial Systems*, Univ. Mich. Press. (1975)
12. S. Jung and B. Moon, The Natural Crossover for the 2D Euclidean TSP, *Genetic and Evolutionary Computation Conference*, pp. 1003-1010. (2000)
13. N. Krasnogor, and J. Smith, A Memetic Algorithm With Self-Adaptive Local Search: TSP as a case study, *Proc. of the Int'l Genetic and Evolutionary Computation Conference - GECCO2000*. (2000)
14. J. J. Grefenstette, Incorporating problem specific knowledge into genetic algorithms, *Genetic Algorithms and Simulated Annealing*, ed. L. Davis, Morgan Kaufmann, Los Altos, CA pp. 42–60. (1987)
15. M. P. P. Larranaga, C. M. H Kuijpers and R. H. Murga, Decomposing bayesian networks: triangulation of the moral graph with genetic algorithms, *Statistics and Computing (UK)*, vol. 7, no. 1, pp. 19–34. (1997)
16. P. Larranaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators, *Artif. Intell. Rev.* 13(2): 129-170. (1999)

17.S. Lin, Computer solutions of the traveling salesman problem, *Bell Syst. Tech. J.*, 44 2245–69. (1965)
18.S. Lin, and B. W. Kernighan, An effective heuristic algorithm for the traveling salesman problem, *Operations Research*, 21:498–516. (1973)
19.Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin: Springer. (1992)
20.P. Moscato, and M. G. Norman, A Memetic Approach for the Traveling Salesman Problem Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems, *Parallel Computing and Transputer Applications*, pp. 177-186. (1992)
21.H. Mühlenbein, M. G. Schleuter, and O. Krämer, Evolution algorithms in combinatorial optimization, *Parallel Computing*, vol. 7, pp. 65–85. (1988)
22.H. Mühlenbein, Parallel genetic algorithms, population genetics and combinatorial optimization, *Proceedings of the Third International Conference on Genetic Algorithms* (J. D. Schaffer, ed.), (San Mateo, CA), Morgan Kaufman. (1989)
23.Y. Nagata and S. Kobayashi, Edge assembly crossover: A high-power genetic algorithm for the travelling salesman problem, *Proc. Of the 7th Int'l Conf. on GAs*. (1997)
24.I. M. Oliver, D. J. Smith, and J. R. C. Holland, A study of permutation crossover operators on the travelling salesman problem, *Genetic algorithms and their applications: Proc. of the second Int. Conf. On Genetic Algorithms* (J. J. Grefenstette, ed.), (Hillsdale, NJ), pp. 224–230, Lawrence Erlbaum Assoc. (1987)
25.E. Ozcan, and C. K. Mohan, Steady State Memetic Algorithm for Partial Shape Matching, *7th Annual Conference on Evolutionary Programming*, pp. 527-536. (1998)
26.E. Ozcan, E. Onbasioglu, Genetic Algorithms for Parallel Code Optimization, *IEEE Congress on Evolutionary Computation*, to appear. (2004)
27.N. J. Radcliffe, and P.D. Surry, Formal memetic algorithms, *Evolutionary Computing: AISB Workshop*, Springer Verlag, LNCS 865, pp. 1-16. (1994)
28.D. Seo and B. Moon, Voronoi Quantized Crossover for Traveling Salesman Problem, *Genetic and Evolutionary Computation Conference*, pp. 544-552. (2002)
29.G. Syswerda, *Schedule optimization using genetic algorithms*, ch. 21, pp. 332–349. (1991)
30.G. Tao and Z. Michalewicz, Inver-over operator for the TSP, *Parallel Problem Solving from Nature – PPSN V* (A. E. Eiben, T. Back, M. Schoenauer, and H.-P. Schwefel, eds.), (Berlin), pp. 803–812, Springer. Lecture Notes in Computer Science 1498. (1998)
31.D. Whitley, T. Starkweather, and D. Fuquay, Scheduling problems and traveling salesman: The genetic edge recombination operator, *Proceedings of the Third International Conference on Genetic Algorithms* (J. D. Schaffer, ed.), (San Mateo, CA), Morgan Kaufman. (1989)
32.G. Ucoluk, Genetic Algorithm Solution of the TSP Avoiding Special Crossover and Mutation, *Intelligent Automation and Soft Computing*, 3(8), TSI Press. (2002)