Mathematical and Computational Applications, Vol. 18, No. 3, pp. 313-322, 2013

A NEW SIMULATED ANNEALING APPROACH FOR TRAVELLING SALESMAN PROBLEM

Hüsamettin Bayram[1] and Ramazan Şahin[2]

[1]Department of Industrial Engineering Hitit University, Çorum, Turkey
[2]Department of Industrial Engineering Gazi University, Ankara, Turkey
husamettinbayram@hitit.edu.tr, rsahin@gazi.edu.tr

Abstract- The aim of this study is to improve searching capability of simulated annealing (SA) heuristic through integration of two new neighborhood mechanisms. Due to its ease of formulation, difficulty to solve and various real life applications several Travelling Salesman Problems (TSP) were selected from the literature for the testing of the proposed methods. The proposed methods were also compared to conventional SA with swap neighborhood. The results have shown that the proposed techniques are more effective than conventional SA, both in terms of solution quality and time.

Key Words- Simulated Annealing, Travelling salesman problem, Roulette wheel selection, Meta-heuristics

1. INTRODUCTION

Meta-Heuristics are optimization techniques that start from an initial solution and search solution space iteratively improving the initial solution. Quality of solutions is improved during the search with regard to a given measure of quality. SA, Genetic Algorithms, Tabu Search are some of the popular nature inspired meta-heuristic algorithms which are used for the solution of combinatorial optimization problems. SA is based on the analogy between annealing of solids and solving of combinatorial optimization problems [1]. The SA algorithm is a stochastic meta-heuristic technique that uses randomized search and randomized acceptance methods which in return provides SA to escape from local minimum. Thus, SA enables effective searching of solution space using its specific mechanisms.

Thanks to the ease of formulation, difficulty to solve and various real life applications TSP is probably the most studied discrete optimization problem [2]. In TSP, number of cities (nodes) and the distances between them are known. The TSP is the problem of finding the shortest route that visits each city exactly once and returns to its origin. The TSP belongs to the class of NP-Hard problems. Therefore, for the large problems, heuristic approaches are the only viable solution techniques.

In this study, we introduce two modified versions of SA with new neighbor solution searching strategies. We tested these new approaches using several TSPs from the literature and we compared them to the conventional SA technique. Improvements will be discussed and the results of the experiments will be given in the following sections.

2. SIMULATED ANNEALING ALGORITHM

Mechanisms within SA prevent the search to quickly converge around a local minimum. During the search, SA not only accepts better solutions but also the worse solutions but with a decreasing probability. The probability of a worse solution to be accepted is determined by two parameters: the temperature and the difference between the objective function values (OFV) of current solution and neighbor solution. The aim of accepting worse solutions is to avoid converging of search to a local minimum. At higher temperatures, the probability of accepting worse solutions is much higher. But, as the temperature decreases, the probability of accepting worse solution decreases. In the following figure algorithmic steps of conventional SA is summarized.

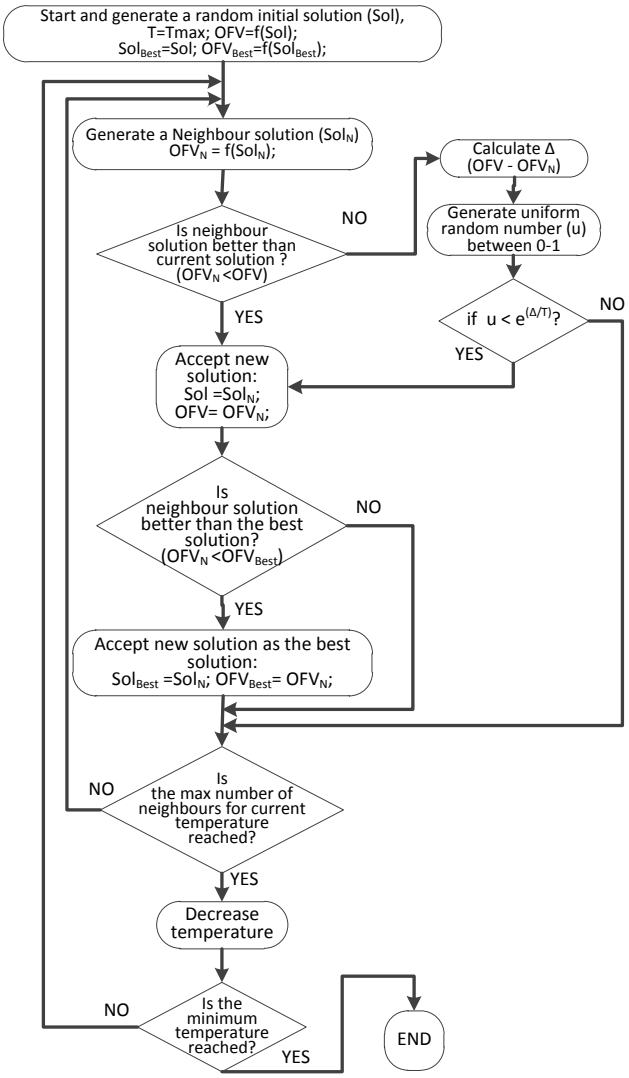


Figure 1. Algorithm of Simulated Annealing

3. LITERATURE REVIEW

SA is a probabilistic meta-heuristic algorithm proposed by [3] and simultaneously by [4]. It has been proven that, SA is capable of solving many real life combinatorial optimization problems including scheduling problems [5], facility layout planning problems [6], assembly line balancing [7], vehicle routing [8] etc. As these studies are based on implementation of SA to a specific problem, some studies that contributes to the algorithmic steps of the SA are discussed in following paragraphs.

[9] proposed a fast SA approach where the cooling schedule of the SA is inversely linear in time. They showed their new cooling strategy is superior to the conventional SA technique. [10] suggested adaptive SA technique in order to reduce user defined search parameters. [11] studied Very Fast SA, where he introduced a new exponentially decreasing cooling schedule. [12] used threshold accepting method, which is principally simpler than conventional SA technique. They demonstrated their technique using TSP and showed that threshold accepting yields very near to optimum results for several known TSPs.

The authors of [13] presented a multi-objective Pareto SA approach, with the aim of finding set of efficient solutions formulti-objective combinatorial optimization problems.Objective weights are employed for the overall evaluation of solutions. [14] discussed parallel SA approach and they tested this technique using several TSPs from the literature. They have found out that the serial implementation of the SA is superior to conventional SA for the solving of TSP.

For the solution of TSP, the authors [15] improved adaptive SA with greedy search and they introduced three different mutation strategies for the generation of new solutions. Thus the convergence of SA is improved compared to several other algorithms in the literature. In [16] authors improved SA with greedy gradient mechanism and they implemented this technique for the solution of a course timetabling problem.

The above mentioned studies are a very little portion of studies that contribute to the SA algorithm. However, these techniques either do not use or partially use problem data in an intelligent and stochastic way for the generation of new solutions. The contribution of this study to the literature is twofold:

1. In order to generate good solutions, the proposed approaches intelligently use problem data or previous search experience by employing Roulette Wheel selection.
2. The proposed techniques do not sacrifice stochasticity and randomness while using problem data, which is essential for successful search of solution space.

4. NEW NEIGHBOUR GENERATION STRAGETY

As stated in previous sections, SA searches solution space by generating new solutions. One of the factors that impact choosing the strategy of neighbor generation is the representation scheme or in other words solution encoding. Representation scheme is the way of representing a candidate solution. In applications of SA to TSP, the most widely used and the simplest representation scheme is permutation encoding. In

permutation encoding the order of the numbers in the array represents the visiting order of the cities. For example, if the third element of the array is "5", then it shows that the third city to be visited is the fifth city.

Selecting two random positions in permutation encoding representation scheme and swapping elements of these positions is the easiest and most widely used way of generating of neighbor solutions. In order to help to understand, in the Figure 2,two neighbor solutions (routes) for a TSP, the permutation encoding representation schemes that correspond to those solutions are given and calculations of their OFVs are described.

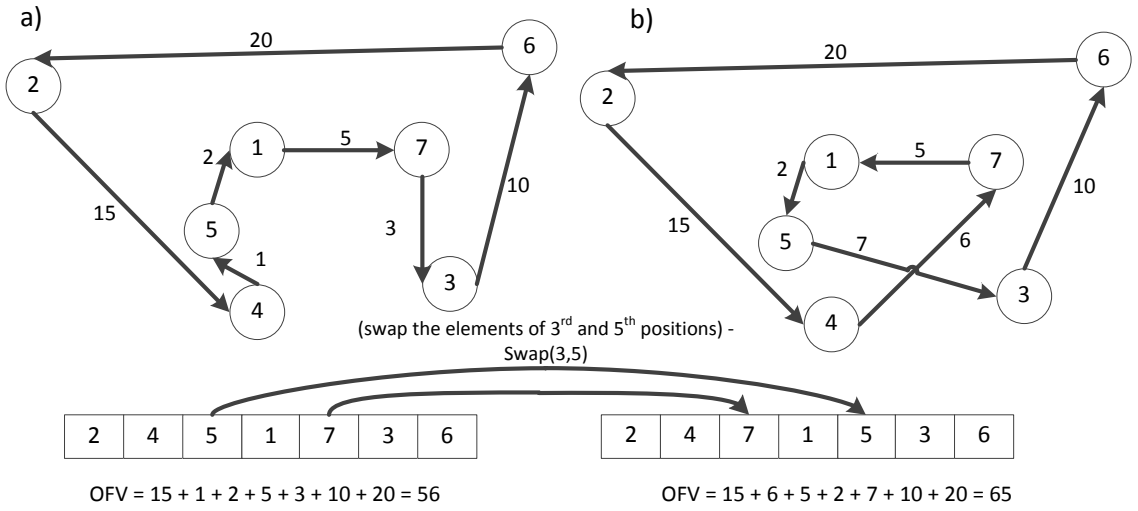
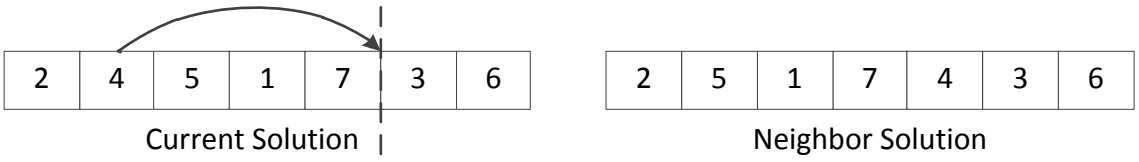


Figure 2. Solution, solution representation and neighbor generation with swap operator (a. current solution, b. neighbor solution)

As stated in previous paragraph, this neighbor searching structure is entirely random and does not take into account the lengths of entering arcs. We assume that the example given in the figure is for a symmetric TSP. In the figure, swap operator is applied to the positions 3 and 5. As it can be seen it the figure, after the swap neighbor generation operator (swap operator) edges 4-5, 5-1, 1-7 and 7-3 left the solution and edges 4-7, 7-1, 1-5 and 5-3 entered to the solution. Swap operator does not take into account the length of the entering edges. Therefore, a neighbor generation technique which intelligently uses the information of arch lengths without deteriorating diversification capability and randomness of SA is needed.

In Genetic Algorithms, Roulette Wheel selection or fitness proportionate selection is a widely used technique for the determination of parent solutions to be combined. As its name implies, in fitness proportionate selection, fitness values are assigned to the candidate solutions and a better fitness value increases the solution's chance to be selected. Namely, every solution is associated with a probability which is proportional to its fitness value (OFV). Similarly, we propose to use Roulette Wheel as a selection mechanism of the edge that will enter to the solution. In the case of TSP, since the goal of TSP is to minimize the total length of route, shorter edges will enter to

the solution with higher probability. Introduction of edges to the solution is described in the following figure.



Let's assume that after performing Roulette Wheel selection, 7-4 is selected as entering edge. Here, 7^{th} node is origin (source), while the 4^{th} node is destination (sink). Firstly, sink node (4^{th} node) is separated from its preceding and succeeding nodes and connected to the source node(7^{th} node) as the sink node. Namely, relative position of source node is saved and sink node is moved to the subsequent position of the source node.

5. PROPOSED METHODS

In this study, we propose two different extensions of SA. In the first method, entering nodes are selected from distance matrix, employing Roulette Wheel selection. In Roulette Wheel, shorter edges are selected with a higher probability. Selected matrices are introduced to the solution following the steps described in the last paragraph of the previous section. In brief, the first method is very similar to the conventional SA but the neighbor solutions will be generated with the help of Roulette Wheel selection.

The second method consists of two stages. In the first stage of this method conventional SA with swap operator is performed. During the first stage two different matrices are recorded. In the first matrix, the number of times each edge is part of a solution is recorded. In the second matrix, total OFV values for each of the edges for the solutions they enter during the search is recorded. At the beginning of the second stage of the second method, a third matrix is calculated for once. In this matrix, average OFV values of every edge are calculated using the first and the second matrices. The second matrix is divided by the first matrix, element by element. A lower average OFV value for an edge is an indicator of that edge can be a part of good solution with a higher probability. These average values are then used for the determination of entering edges in the second stage of the second method, instead of distance matrix. Similar to the first method, Roulette Wheel selection is applied in the second method using the third matrix and entering edges are determined and introduced to the solutions in every neighbor search. The methods are in summarized in below paragraph.

Method 1:

Step 1: Initialization

Step 1.1: Generate a random solution, *Sol*.

Step 1.2: Calculate solution's *OFV*, *OFV=f(Sol)*

Step 1.3:Assign the initial "best solution" and its OFV value; $Sol_{Best} = Sol$; OFV_{Best}= f(Sol_{Best});

Step 1.4: Set initial temperature; $T=T_{max}$.

Step 2: Simulated Annealing

Step 2.1:Select a random edge employing Roulette Wheel selection and using the distance matrix.

Step 2.2: Generate a neighbor solution (Sol_N) by introducing new edge to the matrix and calculate neighbor solution's *OFV*; $OFV_N = f(Sol_N)$.

Step 2.3: If $OFV_N<OFV$ accept new solution and move to the step2.4.Else, generate a uniform random number between 0-1 (u) and calculate $\Delta = OFV - OFV_N$. If $u< e^{\Delta/T}$ accept new solution and move to the step2.4. Else, reject solution and move to the Step2.5.

Step 2.4: Set $Sol = Sol_N$, $OFV - OFV_N$. If $OFV_N<OFV_{Best}$ set $Sol_{Best} = Sol_N$ and $OFV_{Best} = OFV_N$.

Step 2.5: If maximum number of searches reached for this temperature, then decrease the temperature; $T = T \times \lambda$, $(\lambda <1)$.

Step 2.6: If the minimum temperature is reached, stop the search and proceed to the Step3. Else, go to the step 2.1.

Step 3: Report Sol_{Best} and OFV_{Best}.

Method 2:

Step 1: Initialization

Step 1.1: Generate a random solution, *Sol*.

Step 1.2: Calculate solution's *OFV*, $OFV=f(Sol)$

Step 1.3:Assign the initial "best solution" and its OFV value; $Sol_{Best} = Sol$; $OFV_{Best}= f(Sol_{Best})$;

Step 1.4: Set initial temperature; $T=T_{max}$.

Step 2: Simulated Annealing 1

Step 2.1: Select two random positions in the current solution (*Sol*).

Step 2.2: Generate a neighbor solution (Sol_N) by swapping the elements of these positions calculate neighbor solution's *OFV*;$OFV_N = f(Sol_N)$.

Step 2.3: If $OFV_N<OFV$ accept new solution and move to the Step 2.4. Else, generate a uniform random number between 0-1 (u) and calculate $\Delta;\Delta = OFV - OFV_N$. If $u< e^{\Delta/T}$ accept new solution and move to the step 2.4. Else, reject solution and move to the Step2.5.

Step 2.4: Set $Sol = Sol_N$, $OFV - OFV_N$. Increase edge count matrix's elements (matrix 1), that correspond to the edges in S_N, by 1. Namely, if there is an edge i-j in Sol_N, increase the element in the i^{th} row and in j^{th} column by 1. In addition, add OFV_N to the same elements of the matrix 2. If $OFV_N<OFV_{Best}$ set $Sol_{Best} = Sol_N$ and $OFV_{Best} = OFV_N$.

Step 2.5: If maximum number of searches reached for this temperature, then decrease the temperature; $T = T \times \lambda$, $(\lambda <1)$. Else go to step 2.1.

Step 2.6: If the minimum temperature is reached, stop the search and proceed to the Step3. Else, go to the step2.1.

Step 3:Divide matrix 2's every element by same element of the matrix 1 and create a new matrix that stores an average value for every edge (Matrix 3).

Step 4: Initialization

Step 4.1: Generate a random solution, *Sol*.

Step 4.2: Calculate solution's *OFV*, $OFV=f(Sol)$

Step 4.3: Assign the initial "best solution" and its OFV value; Sol_{Best} = Sol; OFV_{Best}= f(Sol_{Best});

Step 4.4: Set initial temperature; T=T_{max}

Step 5: Simulated Annealing

Step 5.1: Select a random edge employing Roulette Wheel selection and using the matrix3.

Step 5.2: Generate a neighbor solution (Sol_N) introducing new edge to the matrix and calculate its length; $OFV_N = f(Sol_N)$.

Step 5.3: If OFV_N<OFV accept new solution and move to the step5.4. Else, generate a uniform random number between 0-1 (u) and calculate Δ; $\Delta = OFV - OFV_N$. If $u < e^{\Delta/T}$ accept new solution and move to the step5.4. Else, reject solution and move to the Step5.5.

Step 5.4: Set $Sol = Sol_N$, $OFV - OFV_N$. If OFV_N<OFV_{Best} set $Sol_{Best} = Sol_N$ and $OFV_{Best} = OFV_N$.

Step 5.5: If maximum number of searches reached for this temperature, then decrease the temperature; $T = T \times \lambda$, (λ <1).

Step 5.6: If the minimum temperature is reached, stop the search and proceed to the step6. Else go to the step5.1.

Step 6: Report Sol_{Best} and OFV_{Best}.

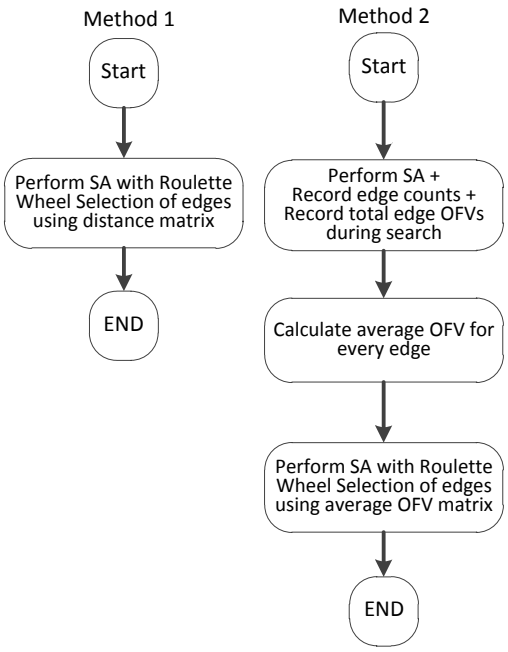


Figure 3. Graphical illustration of the proposed methods

6. EXPERIMENTS

For the testing of these two proposed methods, 11 example problems from the literature were used. Then these two methods are compared to conventional SA with swap operator. Experiments were conducted in two stages. In the first stage of the experiments three methods were tested without time limit and with the same SA parameters. In the second stage, time limited versions of the techniques were run and

compared. All these methods were written in MATLAB programming language and every method was run 30 times for every problem. For the comparison of the techniques, relative error of the average results with respect to the optimal solution was used. Calculation of average error rate is described below: Results of the first stage of the experiments are given in Table 1.

$$\text{Average Error Rate (\%)} = \frac{(\text{average OFV}) - (\text{OFV optimal})}{(\text{OFV optimal})} \times 100 \quad (1)$$

Table 1. First stage of the experiments

		Tmax	Tmin	Epoch	Cooling rate (alpha)	Min	Max	Avg	Std. dev.	# of Solutions searched	Time(avg.)	Avg. Err. (%)	Optimal
Method 1	Gr17[17]	100000	1	20	0.99	2085	2149	2092.5	14.73	22900	0.69	0.36	2085
	Gr21 [17]	100000	1	20	0.99	2707	3000	2738.6	72.69	22900	0.75	1.17	2707
	Gr24 [17]	100000	1	20	0.99	1272	1408	1322.7	37.65	22900	0.75	3.99	1272
	Bayg29 [18]	100000	1	20	0.99	1610	1813	1658.2	53.50	22900	0.80	2.99	1610
	Dantzig42 [19]	100000	1	20	0.999	699	740	713.5	12.99	230140	9.69	2.08	699
	Gr48 [17]	100000	1	20	0.999	5072	5414	5195.2	95.56	230140	10.14	2.96	5046
	Berlin52 [19]	100000	1	20	0.999	7542	8719	8049.2	281.24	230140	11.36	6.73	7542
	St70 [17]	100000	1	20	0.999	678	797	727.9	26.29	230140	13.69	7.84	675
	Pr76 [20]	100000	1	20	0.999	109769	125739	117335.2	4068.60	230140	15.93	8.48	108159
	KroA100 [21]	100000	1	200	0.999	21343	23567	22129.1	573.04	2301400	212.90	3.98	21282
	KroA150 [21]	100000	1	200	0.999	27176	30549	28494.4	782.76	2301400	388.93	7.43	26524
Method 2	Gr17 [17]	100000	1	20	0.99	2085	2103	2089.7	7.05	22900	0.70	0.23	2085
	Gr21 [17]	100000	1	20	0.99	2707	3071	2737.0	85.51	22900	0.73	1.11	2707
	Gr24 [17]	100000	1	20	0.99	1279	1382	1318.0	28.15	22900	0.74	3.61	1272
	Bayg29 [18]	100000	1	20	0.99	1610	1772	1652.7	44.66	22900	0.77	2.65	1610
	Dantzig42 [19]	100000	1	20	0.999	699	753	713.7	15.02	230140	9.87	2.10	699
	Gr48 [17]	100000	1	20	0.999	5055	5326	5144.4	66.98	230140	10.50	1.95	5046
	Berlin52 [19]	100000	1	20	0.999	7542	8261	7948.9	172.86	230140	11.57	5.40	7542
	St70 [17]	100000	1	20	0.999	684	777	718.9	23.13	230140	14.10	6.50	675
	Pr76 [20]	100000	1	20	0.999	110458	124109	114672.7	2505.19	230140	15.19	6.02	108159
	KroA100 [21]	100000	1	200	0.999	21282	23620	21831.9	536.10	2301400	213.03	2.58	21282
	KroA150 [21]	100000	1	200	0.999	27315	28788	28002	426.79	2301400	387.26	5.57	26524
SA with swap operator	Gr17 [17]	100000	1	20	0.99	2085	2210	2129.7	36.53	22900	0.24	2.14	2085
	Gr21 [17]	100000	1	20	0.99	2707	3276	2922.8	197.42	22900	0.26	7.97	2707
	Gr24 [17]	100000	1	20	0.99	1272	1476	1393.6	42.15	22900	0.24	9.56	1272
	Bayg29 [18]	100000	1	20	0.99	1653	1979	1805.2	81.88	22900	0.23	12.12	1610
	Dantzig42 [19]	100000	1	20	0.999	710	879	783.0	40.54	230140	2.37	12.02	699
	Gr48 [17]	100000	1	20	0.999	5054	6361	5674.5	261.67	230140	2.24	12.46	5046
	Berlin52 [19]	100000	1	20	0.999	7872	9249	8469.2	301.10	230140	2.356	12.29	7542
	St70 [17]	100000	1	20	0.999	782	957	868.6	37.98	230140	2.20	28.68	675
	Pr76 [20]	100000	1	20	0.999	122859	147863	136155.8	6023.71	230140	2.43	25.88	108159
	KroA100 [21]	100000	1	200	0.999	23677	27760	25597.7	1157.55	2301400	21.89	20.28	21282
	KroA150 [21]	100000	1	200	0.999	31744	37599	34832.7	1315.70	2301400	21.67	31.33	26524

As it can be seen in Table 1, in the first stage of the experiments the results obtained using first and second methods are better compared to the results reached using conventional SA with swap operator. However, first and second methods' running time was up to 18 times longer than that of conventional SA due to the computational complexity. In the second stage of the experiments, conventional SA's epoch length is increased in such a way that its running time slightly exceeds that of first and second methods. The aim of that is to compare proposed methods' effectiveness in a limited time. For this reason, in the second stage of experiments only conventional SA was run with increased epoch length. The results of the second stage of the experiment are given in Table *2*.

Table 2. Second stage of the experiments

		Tmax	Tmin	Epoch	Cooling rate (alpha)	Min	Max	Average	Standard deviation	# of Solutions searched	Time (avg.)	Avg. Err. (%)	Optimal
SA with swap operator	Gr17 [17]	100000	1	60	0.99	2085	2136	2104.9	17.65	68700	0.70	0.95	2085
	Gr21 [17]	100000	1	60	0.99	2707	3254	2828.5	159.06	68700	0.76	4.49	2707
	Gr24 [17]	100000	1	65	0.99	1272	1432	1340.9	47.10	74425	0.76	5.42	1272
	Bayg29 [18]	100000	1	70	0.99	1620	1808	1700.5	57.55	80150	0.81	5.62	1610
	Dantzig42 [19]	100000	1	85	0.999	699	784	741.6	23.48	978095	10.09	6.09	699
	Gr48 [17]	100000	1	95	0.999	5071	5709	5373.3	202.66	1093165	11.03	6.49	5046
	Berlin52 [19]	100000	1	99	0.999	7670	8613	8194.8	248.10	1139193	12.52	8.66	7542
	St70 [17]	100000	1	129	0.999	704	813	757.5	27.48	1484403	14.20	12.22	675
	Pr76 [20]	100000	1	150	0.999	113943	132597	122147.2	4150.12	1726050	16.03	12.93	108159
	KroA100 [21]	100000	1	2050	0.999	21715	24693	23295.8	738.29	23589350	235.94	9.46	21282
	KroA150 [21]	100000	1	3650	0.999	27869	30931	29541.6	871.87	41296044	395.33	11.38	26524

In the second stage of the experiments, conventional SA with swap operator was performed. In order to increase its running time and improve the quality of solutions, epoch lengths were increased. As it can be seen in Table *1* and Table *2*, results obtained using the proposed methods are superior to the results obtained using conventional SA with swap operator. It can be inferred that, neighbor generation through intelligent introduction of edges increases the capability of SA to reach better solution. In addition, second method was also slightly better than the first method. But, it must be kept in mind that, the times given for second method are not the actual running times of second method. The time required for the generation of matrix 1 and matrix 2 are excluded. That means, it cannot be asserted that second method is better than first method.

7. CONCLUSION

This paper introduces two new solution approaches for TSP based on SA. These techniques employ Roulette Wheel selection strategy for the selection of entering edges to a solution representation. The first method uses distance matrix and the second method uses a new matrix which is created using the average OFVs of previously visited solutions. Then, the proposed methods were tested on 11 well known benchmark problem sets from literature. After the experiments, it is found out that the proposed methods are superior to conventional SA with swap parameter in reaching optimal and suboptimal solutions. The main advantage of these techniques is improved convergence capability by using problem data and without losing inherent stochasticity in the SA technique. The paper has also revealed that the Roulette wheel selection of genetic algorithm is also applicable in SA with success.

8. REFERENCES

1. E. Aarts, J. K. Lenstra, *Local search in combinatorial optimization*, Chichester, U.K., Wiley.

2. G. Gutin, Traveling salesman problems, *Handbook of Graph theory*, Boca Raton, U.S.A, CRC Press, 2003.

3. S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by Simulated Annealing, *Science* **220**, 671–680, 1983.

4. V. Cerny, Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm, *Journal of Optimization Theory and Applications* **45(1),** 41–51, 1985.
5. R. Zhang, C. Wu, A hybrid immune simulated annealing algorithm for the job shop scheduling problem, *Applied Soft Computing* **10**, 79–89, 2010.
6. R. Şahin, A Simulated Annealing Algorithm for Solving the Bi-Objective Facility Layout Problem, Expert Systems with Applications **38(4),** 4460–4465, 2011.
7. B. Cakir, F. Altiparmak, B. Dengiz, Multi-objective optimization of a stochastic assembly line balancing: A hybrid simulated annealing algorithm, *Computers & Industrial Engineering* **60**, 376–384, 2011.
8. S.W. Lin, V.F. Yu, S.Y. Chou, Solving the truck and trailer routing problem based on a simulated annealing heuristic, *Computers & Operations Research* **36**, 1683–1692, 2009.
9. H. Szu, R. Hartley, Fast Simulated Annealing, *Physics Letters A* **122**, 157-162, 1987.
10. L. Ingber, A. Petraglia, M.R. Petraglia, M.A.S. Machado, Adaptive simulated annealing, *Stochastic global optimization and its applications with fuzzy adaptive simulated annealing*, 33-62, Springer Berlin Heidelberg.
11. L. Ingber,Very fast simulated re-annealing, *Mathematical and computer modeling* **12**, 967-973, 1989.
12. G. Dueck, T. Scheuer, Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing, *Journal of computational physics* **90**, 161-175, 1990.
13. P. Czyzżak, A. Jaszkiewicz, Pareto simulated annealing-a metaheuristic technique for multiple-objective combinatorial optimization, *Journal of Multi-Criteria Decision Analysis* **7**, 34-47, 1998.
14. M. Malek, M. Guruswamy, M. Pandya, H. Owens, Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem, *Annals of Operations Research* **21**, 59-84, 1989.
15. X. Geng, Z. Chen, W. Yang, D. Shi, K. Zhao, Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search, *Applied Soft Computing* **11**, 3680-3689, 2011
16. M. Kalender, A.Kheiri, E. Ozcan, E.K. Burke, A greedy gradient-simulated annealing hyper-heuristic for a curriculum-based course timetabling problem, In Computational Intelligence (UKCI), 2012 12th UK Workshop on (pp. 1-8). IEEE, 2012.
17. M. Grötschel and O. Holland, Solution of Large Scale Symmetric Travelling Salesman Problems, *Mathematical Programming* **51(1-3),**141–202,1991.
18. M. Grötschel, *Optimierungsmethoden I*, Lecture Notes, University of Augsburg.
19. G. B. Dantzig, D. R. Fulkerson, S. M. Johnson, Solution of a Large Scale Traveling-Salesman Problem, *Operations Research* **2**, 393–410, 1954.
20. M. W. Padberg, G. Rinaldi, A Branch and Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems, *IASI Research Report* **247**, 1988.
21. W. Felts, P. Krolak, G. Marble, A Man-Machine Approach towards Solving the Travelling-Salesman Problem, *Communications ACM* **14**, 327–334, 1971.