International Journal of Industrial Engineering & Production Research

March 2011, Volume 22 Number 1
pp. 31-42

http://IJIEPR.iust.ac.ir/

ISSN: 2008-4889

Finding the Shortest Hamiltonian Path for Iranian Cities Using Hybrid Simulated Annealing and Ant Colony Optimization Algorithms

M. Yaghini*, M. Momeni, M. Sarmadi

**M. Yaghini*, Assistant Professor, School of Railway Engineering, Iran University of Science and Technology, Tehran, Iran*
M. Momeni, MSc., School of Railway Engineering, Iran University of Science and Technology, Tehran, Iran
M. Sarmadi, MSc., School of Railway Engineering, Iran University of Science and Technology, Tehran, Iran

KEYWORDS

Iranian cities,
Traveling salesman problem,
Hamiltonian path,
Simulated annealing algorithm,
Ant colony optimization algorithm

ABSTRACT

The traveling salesman problem is a well-known and important combinatorial optimization problem. The goal of this problem is to find the shortest Hamiltonian path that visits each city in a given list exactly once and then returns to the starting city. In this paper, for the first time, the shortest Hamiltonian path is achieved for 1071 Iranian cities. For solving this large-scale problem, two hybrid efficient and effective metaheuristic algorithms are developed. The simulated annealing and ant colony optimization algorithms are combined with the local search methods. To evaluate the proposed algorithms, the standard problems with different sizes are used. The algorithms parameters are tuned by design of experiments approach and the most appropriate values for the parameters are adjusted. The performance of the proposed algorithms is analyzed by quality of solution and CPU time measures. The results show high efficiency and effectiveness of the proposed algorithms.

1. Introduction

The Traveling Salesman Problem (TSP) is a well known and important combinatorial optimization problem. The goal of this problem is to find the shortest path that visits each city in a given list exactly once and then returns to the starting city. The distances between n cities are stored in a distance matrix **D** with elements d_{ij} where $i, j = 1, 2, ..., n$ and the diagonal elements d_{ij} are zero [1]. In the 1920's, the mathematician and economist Menger publicized this problem among his colleagues in Vienna. In the 1930's the problem reappeared in the mathematical circles of Princeton [2]. In the 1940's, mathematician Flood publicized the name, TSP, within the mathematical community at mass [3].

Despite this simple problem statement, solving the TSP is difficult, since it belongs to the class of NP-complete problems [4]. Different instances of the TSP are divided into different classes based on the arrangement of distance between the cities or the type of graph in concern. In the symmetric TSP, the distance between two cities is the same in each direction, forming an undirected graph. The importance of the TSP arises besides from its theoretical appeal from the variety of its applications.

Typical applications in operations research include vehicle routing, computer wiring, cutting wallpaper, job sequencing, and job scheduling. Recognizing methods to solve TSP and find the appropriate method is one of the important research areas [1] and [5]. Due to differences between metaheuristics algorithms, Simulated Annealing (SA) algorithm as a single-solution-based algorithm and Ant Colony Optimization (ACO) algorithm as a population-based-algorithm are used to find the shortest Hamiltonian path between 1071 Iranian cities. The algorithms parameters are tuned by Design of Experiments (DOE) approach and the most appropriate values for the parameters are adjusted. To evaluate the accuracy and efficiency of

* **Corresponding author. M. Yaghini**
Email: *yaghini@iust.ac.ir*
Paper first received May. 15. 2010, and in revised form Dec. 05. 2010.

the proposed algorithms, standard problems with size from 16 to 1060 cities are used. The paper is organized as follows. In section two, literature review is conducted. In the third section, the proposed hybrid SA and ACO algorithm are presented. In the fourth section, parameters tuning, and in the fifth section the result of the proposed algorithms are discussed. In section six, the statistical analysis is presented. The shortest Hamiltonian path between Iranian cities is derived by using the proposed algorithms in section seven. Conclusions are presented in section eight.

2. Literature Review

The TSP problem has different types, such as Probabilistic Traveling Salesman Problem (PTSP), Traveling Salesman Problem with Time Windows (TSPTW), Multicommodity Traveling Salesman Problem (MTSP), and Railway Traveling Salesman Problem (RTSP). The PTSP is a topic of theoretical and practical importance in the study of stochastic network problems. It provides researchers with a modeling framework for exploring the stochastic effects in routing problems. Designing effective algorithms for stochastic routing problems is a difficult task. This is due to the element of uncertainty in the data, which increases the difficulty of finding an optimal solution in a large search space [6] and [7]. The TSPTW involves finding the minimum cost tour in which all cities are visited exactly once within their requested time windows. This problem has a number of important practical applications including scheduling and routing [8].

The MTSP presents a more general cost structure, allowing for solutions that consider the quality of service to the customers, delivery priorities and delivery risk, among other possible objectives. In the MTSP, the salesman pays the traditional TSP fixed cost for each arc visited, plus a variable cost for each of the commodities being transported across the network [9] and [10]. The RTSP in which a salesman using the railway network to visit a certain number of stations to carry out business, starting and ending at the same station, and having as goal to minimize the overall time of the journey. It is related to the Generalized Asymmetric Traveling Salesman Problem (GATSP) [11], [12], [13], and [14]. TSP solution methods can be divided into two groups including exact methods and approximation methods. Although exact algorithms exist for solving the TSP, like enumeration method, branch and bound and dynamic programming but it has been proved that for large-size TSP, it is almost impossible to generate an optimal solution within a reasonable amount of time. Exact techniques can solve only small instances to optimality.

The approximation methods are classified into metaheuristic and heuristic techniques. Metaheuristics, instead of exact algorithms, are extensively used to solve such problems [15]. For constructing and improving initial solutions in metaheuristics, the heuristic methods such as nearest neighbor, 2-opt, and 3-opt methods can be used [16]. Attempts to solve the TSP were futile until the mid-1950's when Dantzig et al. [17] and [18] presented a method for solving the TSP. They showed the effectiveness of their method by solving a 49-city instance. Radharamanan (1986) developed a branch and bound model with penalty tour building for solving TSP and transportation routing problems [19]. Pop et al. (2008) used a cutting plane approach to solve the railway travelling salesman problem [20]. Sarubbi et al. (2008) presented a branch and cut algorithm for the MTSP [11]. Zamani and Lau (2010) presented an effective procedure that finds lower bounds for the travelling salesman problem based on the 1-tree using a learning-based Lagrangian relaxation technique [21].

Following, the metaheuristic methods for solving TSP are reviewed. Cheng and Gen (1994) described a new crossover operator for the TSP problem, the greedy selection crossover operator, which is designed for path representation and performed at gene level. It can utilize local precedence and global precedence relationship between genes to perform intensive search among solution space to reproduce an improved offspring [22]. Moon et al. (2002) used an efficient genetic algorithm to solve the TSP with precedence constraints.

The key concept of the proposed algorithm is a topological sort, which is defined as an ordering of vertices in a directed graph [23]. Bontoux et al. (2009) solved the GTSP by using memetic algorithm. In this problem, the set of cities is divided into mutually exclusive clusters. The objective of the GTSP consists in visiting each cluster exactly once in a tour, while minimizing the sum of the routing costs [24] and [25].

Balaprakash (2009) customized two metaheuristics, an iterated local search algorithm and a Memetic algorithm to solve the PTSP [26]. Liu (2010) proposed three initial solution generators under a genetic algorithm framework for solving the PTSP. This paper used a set of numerical experiments based on heterogeneous and homogeneous PTSP instances to test the effectiveness and efficiency of the proposed algorithms [27].

Marinakis et al. (2010) proposed a new hybrid algorithmic nature inspired approach based on Particle Swarm Optimization (PSO), Greedy Randomized Adaptive Search Procedure (GRASP), and Expanding Neighborhood Search (ENS) strategy for the solution of the PTSP [28]. Li et al. (2008) presented an improved ant colony optimization algorithm for traveling salesman problem, which adopts a new probability selection mechanism by using Held-Karp lower bound to determine the trade-off between the influence of the heuristic information and the pheromone trail [29]. The ant colony optimization algorithm usually falls into local optimal solution and cannot select the path with high pheromone

concentration quickly in solving the TSP. Tiankun et al. (2009) proposed an improved MAX-MIN Ant System (MMAS) algorithm based on pheromone concentration reinitialization to overcome this problem [30]. Zhang (2009) presented a paper that is a survey of natural computation for the traveling salesman problem [31]. The work of Ghoseari and Sarhadi (2009) is similar to this paper. They tried to find the shortest Hamiltonian path for 360 Iranian cities using ant colony optimization algorithm [32]. The weaknesses of their work are: (1) they did not use the standard problems to evaluate their algorithm, (2) the parameter tuning method was not an appropriate approach, and (3) the solving CPU time (5523 seconds) was very long. In our paper, the number of cities is increase to 1071 cities, the design of experiments approach is used for parameter tuning, the standard problems with different dimensions are used to evaluate the proposed algorithms, and the solving CPU time is decreased dramatically.

3. The Proposed Algorithms

3.1. The Simulated Annealing Algorithm

Simulated annealing algorithm has been introduced in 1983 by Kirkpatrick et. al [33]. This memoryless metaheuristic is base on Metropolis algorithm and prevents remaining in the local optimal with the transition towards low quality solution. The SA algorithm, which is a single-solution-based metaheuristic, has very high ability to be combined with a heuristic algorithm.

The components of the proposed SA algorithm are solution representation, initial solution, neighbourhood structure, initial temperature, cooling schedule, and termination criteria.

To represent solution, an array that its length is the number of cities is used. In each element of this array, the number of a specific city is stored (Fig. 1). Finally, the sequence of cities that located in this array is indicating the constructed path. To create the initial solution, a nearest neighbor heuristic method is used. Using this algorithm leads to create appropriate initial solution and speed up the trend of algorithm toward the optimal solution.

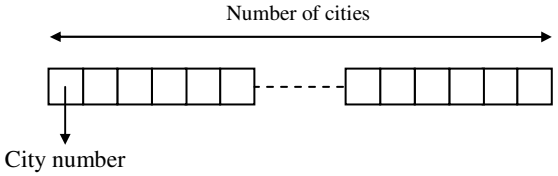


Fig. 1. Solution representation

One of the main components of the SA algorithm is neighborhood structure. Common methods such as inversion, insertion, and swap are not suitable for large-scale problems [34]. Therefore, to solve such problems using local search is appropriate. In this way, for every move, several cities selected randomly and replacement with a determined number of nearest cities are reviewed. Ultimately, the best city is selected to substitute.

This approach leads to better solution quality and reduce CPU time to reach the best solution [35] and [36]. Initial temperature has a large influence on the performance of the proposed algorithm. High initial temperature leads to low performance and running time of algorithm is increased. Low initial temperature does not search solution space well and it cannot escape from local optimal. In parameters tuning, the appropriate amount of the initial temperature is adjusted. In the proposed algorithm for cooling schedule, the linear and geometric methods are used. In parameters tuning section, with considering the quality of solution, an appropriate function is selected for reducing temperature. In the proposed algorithm, the number of iterations without improvement and the final temperature are considered as termination conditions. In the parameter setting, the type of termination condition is determined. Fig. 2 displays pseudocode for the proposed SA algorithm.

Pseudocode for the proposed Simulated Annealing Algorithm

Read data
Let *currentTour* = initial feasible solution by nearest neighbor heuristic method
Let *currentTourLength*= current tour length
Let *bestSoFarTour*= *currentTourLength*
Let *bestSoFarTourLength*= *currentTourLength*
While (stopping criterion not met)
  Let *neighbourTour*= best neighbourhood of current solution
  randomNumber= random number (0,1)
  ΔC = *neighbourTourLength - currentTourLength*
  If ($e^{-\Delta C / initialTemprature} > randomNumber$) **Then**
    Let *currentTour*= *neighbourTour*
    Let *currentTourLength*= *neighbourTourLength*
  End If
  If (current*TourLength* < *bestSoFarTourLength*) **Then**
    bestSoFarTourLength= *currentTourLength*
    bestSoFarTour= *currentTour*
  End If
  initialTemprature= **Apply** cooling function
End While
Output *bestSoFarTourLength*
End.

Fig. 2. Pseudo code for the proposed SA algorithm to find shortest Hamiltonian path

3.2. The Proposed Ant Colony Optimization Algorithm

Dorigo proposed a common framework for the applications and algorithmic variants of a variety of ant

algorithms. Algorithms that fit into the Ant Colony Optimization (ACO) metaheuristic framework called in the following ACO algorithms [37] and [38].

The ACO is inspired by the foraging behavior of ant colonies. The ACO is a population-based algorithm, which is a powerful algorithm for solving combinatorial problems. One of the ACO algorithms is Ant Colony System (ACS) in which, pheromone evaporation is interleaved with tour construction [39] and [40].

In this paper ASC algorithm combined with local search method. The components of proposed ACO algorithm are solution representation, initial pheromone value, number of ants, q_0, α , β, the local evaporation coefficient ζ, public evaporation coefficient ρ, and the termination condition [38]. A first implementation of an ACO algorithm can be quite straightforward.

In fact, if a greedy construction procedure like a nearest-neighbour heuristic is available, one can use as a construction graph. The same graph used by the construction procedure, and then it is only necessary to add pheromone trail variables to the construction graph and define the set of artificial ants.

If the initial pheromone values are too low, then the search is quickly biased by the first tours generated by the ants, which in general leads toward the exploration of inferior zones of the search space. On the other side, if the initial pheromone values are too high, then many iterations are lost waiting until pheromone evaporation reduces enough pheromone values, so that pheromone added by ants can start to bias the search. In order to have an efficient implementation, often additional data structures are required, like arrays to store information which, although redundant, make the processing much faster e.g. candidate list.

The parameter ρ is used to avoid unlimited accumulation of the pheromone trails and it enables the algorithm to forget bad decisions previously taken. The heuristic value is used by the ants' heuristic rule to make probabilistic decisions on how to move on the graph. α and β are two parameters which determine the relative influence of the pheromone trail and the heuristic information. With probability q_0 the ant makes the best possible move as indicated by the learned pheromone trails and the heuristic information (in this case, the ant is exploiting the learned knowledge); while with probability (1- q_0) it performs a biased exploration of the arcs. In the proposed algorithm only one ant (the best-so-far ant) is allowed to add pheromone after each iteration. In addition to the global pheromone trail updating rule, in this algorithm the ants use a local pheromone update rule that they apply immediately after having crossed an arc. In fact, the best trade-off between solution quality and computation time seems to be obtained when using a small number of ants between two and ten. In the proposed algorithm, after making the tour by the ants in a colony, Ant that has produced the best answer, update the pheromone on the possible arcs and increases the rate of pheromone. Once all the ants have terminated their ant solution construction procedure, a pheromone update phase is started in which pheromone trails are modified. Termination condition in the proposed algorithm is the number of iterations without improvements that is specified in parameters tuning section. The proposed ACO algorithm pseudocode is shown in Fig. 3.

```
Pseudocode for proposed Ant Colony Optimization algorithm
Read data
Find candidate list
While (stopping criterion not met)
  For each ant Do
    Construct solution
    Compute tour length
    Local pheromone update
    Global pheromone update
    If (tourLength < bestSoFarTourLength) Then
      bestSoFarTourLength= tourLength
    End If
  End For
End While
Output bestSoFarTourLength
End.
```

Fig. 3. Pseudo code for proposed ACO algorithm to find shortest Hamiltonian path

4. Parameters Tuning

In this section, DOE method, selecting standard problems and parameters tuning of the proposed algorithms are discussed.

The parameters of the proposed algorithms are tuned using Design of Experiments (DOE) approach and Design-Expert statistical software [41]. We can define an experiment as a test or series of tests in which purposeful changes are made to the input variables of a process or system so that we may observe and identify the reasons for changes that may be observed in the output response. DOE refers to the process of planning the experiment so that appropriate data that can be analyzed by statistical methods will be collected, resulting in valid and objective conclusions [42].

The three basic principles of DOE are replication, randomization, and blocking. By replication, we mean a repetition of the basic experiment. Replication has two important properties. First, it allows the experimenter to obtain an estimate of the experimental error. Second, if the sample mean is used to estimate the effect of a factor in the experiment, replication permits the experimenter to obtain a more precise estimate of this effect. By randomization, we mean that both the allocation of the experimental material and the order in which the individual runs or trials of the experiment are to be performed are randomly determined. Randomization usually makes this assumption valid. Blocking is a design technique used to improve the precision with which comparisons

among the factors of interest are made. Often blocking is used to reduce or eliminate the variability transmitted from nuisance factors; that is, factors that may influence the experimental response but in which we are not directly interested [43].

The important parameters in DOE approach are response variable, factor, level, treatment and effect. The response variable is the measured variable of interest. In the analysis of metaheuristics, the typically measures are the solution quality and solution time [44]. A factor is an independent variable manipulated in an experiment because it is thought to affect one or more of the response variables.

The various values at which the factor is set are known as its levels. In metaheuristic performance analysis, the factors include both the metaheuristic tuning parameters and the most important problem characteristics [45]. A treatment is a specific combination of factor levels.

The particular treatments will depend on the particular experiment design and on the ranges over which factors are varied. An effect is a change in the response variable due to a change in one or more factors [46] and [47]. Design of experiments is a tool that can be used to determine important parameters and interactions between them. Four-stages of DOE consist of screening and diagnosis of important factors, modeling, optimization and assessment. This methodology is called sequential experimentation which is used to set the parameters in the DOE approach and has been used in this paper for the proposed algorithms [43].

To adjust the parameters of the proposed algorithms, four standard problems from TSPLIB website has been used [48]. These problems based on the size and dimensions are classified into two groups. For each of problem groups, two blocks are considered. Problems ulysses22 and eil51 as first group block and problems gr202 and pcb442 as second group block are selected (Tab. 1). For each of two groups, parameters tuning has been done separately.

Tab. 1. Standard problem for parameters tuning

First group			Second group		
Problem	**Number of cities**	**Distance type**	**Problem**	**Number of cities**	**Distance type**
ulysses22	22	Geographic	gr202	202	Geographic
eil51	51	Euclidean	pcb442	442	Euclidean

In the proposed SA algorithm, solution quality and CPU time are considered as the response variables. Factors and their levels are shown in Tab. 2. Each block is considered with 16 treatments and main effects. Fig. 4 shows the relationship among relative gap, initial temperature, and cooling rate parameters for the first group problems. The relative gap is derived from Eq. (1). In Tab. 3 the final parameters for solving problems of first and second groups are shown. These parameters are fixed to solve the test problems.

$$relative\ gap = \frac{best\ so\ far\ tour\ length\ -\ optimal\ value}{optimal\ value} \quad (1)$$

Tab. 2. Level factorial design for SA proposed algorithm

	First group		Second group	
Factor	**Low**	**High**	**Low**	**High**
initial temperature	8000	16000	8000	18000
cooling function	Geometric	Linear	Geometric	Linear
cooling rate	0.01	0.9999	0.01	0.9999
final temperature	0.01	0.0001	0.01	0.0001
candidate list rate	0.5	1	0.01	0.1

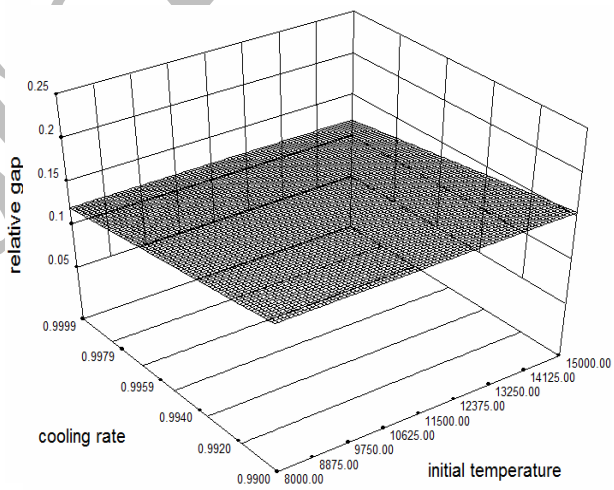


Fig. 4. The relationship of initial temperature, and cooling rate for the first group problem

Tab. 3. Final values of parameters for the SA proposed algorithm

Factor	**First group**	**Second group**
initial temperature	15000	15000
cooling function	Geometric	Geometric
cooling rate	0.9999	0.9999
final temperature	0.0001	0.0001
candidate list rate	0.6	0.01

In proposed ACO algorithm, the quality of solution and CPU time are selected as the response variables. Factors and their levels are shown in Tab. 4. Each block is considered with 64 treatments and main effects. Fig. 5 as an example, Shows the changes of relative gap, toward the changes of α and β parameters for the first group problems. In Tab. 5 the final parameters for solving problems of first and second groups are shown. These parameters to solve the final problems are fixed.

Tab. 4. level factorial design for ACO proposed algorithm

Factor	First group		Second group	
	Low	High	Low	High
α	5	20	2	20
β	5	25	5	25
q_0	0.01	0.5	0.7	1
local evaporate	0.01	0.5	0.08	0.3
global evaporate	0.01	0.5	0.08	0.3
number of ants	10	50	15	30
candidate list rate	0.01	1	0.01	0.05
iteration number	100	500	300	500
initial pheromone rate	1	3	1	2

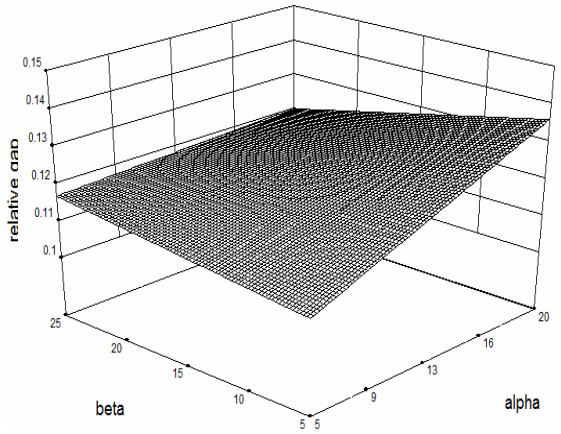


Fig. 5. Change of relative gap into α and β for the first group problem

Tab. 5. Final values of factors for the ACO proposed algorithm

Factor	First group	Second group
α	5	3
β	5	10
q_0	0.5	0.3
local evaporate	0.01	0.3
global evaporate	0.01	0.09
number of ants	50	24
candidate list rate	0.8	0.05
iteration number	500	300
initial pheromone rate	1	1.5

5. The Performance Analysis of the Proposed Algorithms

To evaluate the proposed algorithms, the standard problems with size of 16 to 1060 cities from TSPLIB website are used [48]. The global optimal solutions of these problems are available in this website and are used to analyze the performance of the proposed algorithms. According to direct relationship between CPU time and problem size, the selected problems are divided into two groups. The first group includes the problem with less than or equal 100 cities and the second group includes more than 100 cities. This classification is shown in Tab. 6.

Tab. 6. Standard problems characteristics

First group			Second group		
Problem	Number of cities	Distance type	Problem	Number of cities	Distance type
ulysses16	16	Geographic	a280	280	Euclidean
berlin52	52	Euclidean	gr666	666	Geographic
kroA100	100	Euclidean	pr1002	1002	Euclidean
rd100	100	Euclidean	u1060	1060	Euclidean

Each of the standard problems run ten times and the best solution, the worst solution, the average of solutions, the average of CPU time, the average of relative gaps, and the relative gap of the best solution are achieved. The java programming language is used to implement algorithms. The program was run on a personal with core2 CPU at 2.66 GHz, 4 Gigabytes of Ram, and operating under Microsoft windows vista.

The computational results of the proposed SA algorithm for standard problems are shown in Tab. 7. These results are achieved from ten runs for each problem.

The averages of relative gaps, CPU time, and the best solution relative gap are 10 percent, 22 seconds and 7 percent, respectively. These values indicate that the proposed SA algorithm can has high accuracy and efficiency to solve problems with different dimensions:

Tab. 7. Performance analysis of the proposed SA algorithm

Problem	Global optima	Average of tour length	Average of relative gap	CPU time (s)	Worst solution	Best solution	Best solution relative gap
ulysses16	6859	6859.000	0.000	0.45	6859.00	6859.00	0.0000
berlin52	7542	7914.386	0.049	3.73	8148.42	7544.32	0.0003
kroA100	21282	21923.690	0.030	11.23	22843.53	21356.24	0.0035
rd100	7910	8277.029	0.046	13.64	8436.66	8095.00	0.0234
a280	2579	2985.528	0.158	16.20	3078.69	2809.88	0.0895
gr666	294358	339243.800	0.153	75.55	343562.00	334523.00	0.1364
pr1002	259045	310376.770	0.198	27.07	315063.52	302859.00	0.1691
u1060	224094	268281.650	0.197	31.81	279481.30	261370.80	0.1663
Total average	**-**	**-**	**0.104**	**22.46**	**-**	**-**	**0.0736**

Tab. 8. Performance analysis of the ACO proposed algorithm

Problem	Global optima	Average of tour length	Average of relative gap	CPU time (s)	Worst solution	Best solution	Best solution relative gap
ulysses16	6859	6859.000	0.000	0.98	6859.00	6859.00	0.0000
berlin52	7542	7578.786	0.005	11.09	7716.65	7544.32	0.0003
kroA100	21282	21473.444	0.009	40.62	21709.46	21321.83	0.0018
rd100	7910	8091.455	0.023	11.12	8268.31	7934.84	0.0031
a280	2579	2766.750	0.073	10.30	2807.09	2726.45	0.0572
gr666	294358	331268.600	0.125	57.14	334179.00	329362.00	0.1189
pr1002	259045	298397.400	0.152	127.14	305836.53	294381.00	0.1364
u1060	224094	263901.040	0.178	143.43	269804.99	260785.00	0.1637
Total average	**-**	**-**	**0.071**	**50.23**	**-**	**-**	**0.0602**

The computational results for the ACO algorithm for standard problems are shown in Tab. 8. In this algorithm, the average of relative gap is 7 percent and CPU time is 50 seconds in Tab. 8. The results in this table show the efficiency and effectiveness of the ACO algorithm. After solving the standard problems with proposed algorithms, the average of relative gap, CPU time and best solution relative gap are shown in Tab. 9. Final results are visually compared on Fig. 6, Fig. 7 and Fig. 8. According to the ten runs for each problem and calculate the total average, comparison is very accurate. Effectiveness and efficiency of the algorithms in solving various problems are evaluated by quality of solution and CPU time measures. In Tab. 9, the proposed algorithms are compared based on best solution relative gap, average of relative gap, and CPU time. The performance analysis of two proposed metaheuristic algorithms for solving the standard problems indicating three percent difference in the average of relative gaps and thirty seconds in CPU time. Difference in the best solution relative gap of the proposed algorithms is less than 2 percent. As in Fig. 6, Fig. 7 and Fig. 8 are shown, when the dimensions of the problem become larger, the relative gap and CPU time are increased.

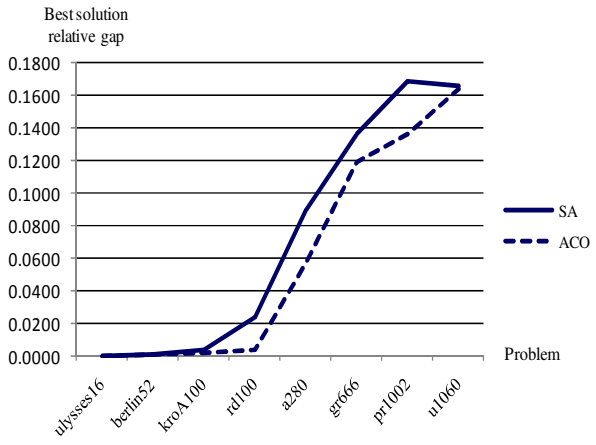
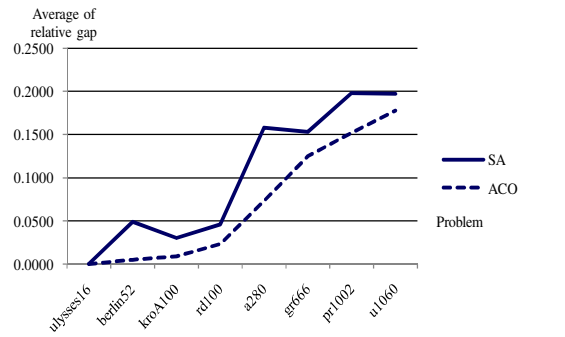


Fig. 6. Comparison of the proposed algorithms in the best solution relative gap



F

Fig. 7. Comparison of the proposed algorithms in the average of relative gap

Tab. 9. Comparison of the proposed algorithms

	Best solution relative gap		Average of relative gap		CPU time (s)	
Problem	**SA**	**ACO**	**SA**	**ACO**	**SA**	**ACO**
ulysses16	0.0000	0.0000	0.000	0.000	0.45	0.98
berlin52	0.0003	0.0003	0.049	0.005	3.73	11.09
kroA100	0.0035	0.0018	0.03	0.009	11.23	40.62
rd100	0.0234	0.0031	0.046	0.023	13.64	11.12
a280	0.0895	0.0572	0.158	0.073	16.2	10.3
gr666	0.1364	0.1189	0.153	0.125	75.55	57.14
pr1002	0.1691	0.1364	0.198	0.152	27.07	127.14
u1060	0.1663	0.1637	0.197	0.178	31.81	143.43
Average	**0.0736**	**0.0602**	**0.104**	**0.071**	**22.46**	**50.23**

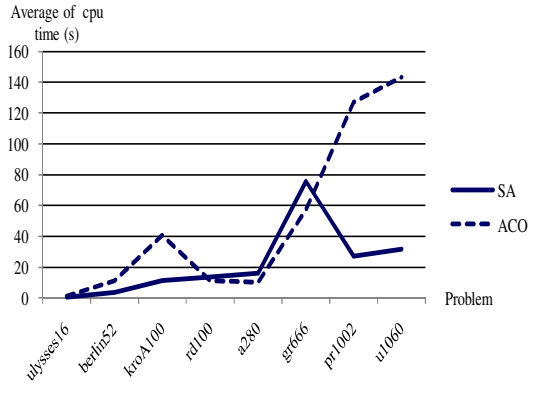


Fig. 8. Comparison of the proposed algorithms in CPU time

As Tab. 9 and Fig. 6 shown, ACO is better than SA algorithm in comparing with the best solution relative gap. With aspect of the average of relative gap, ACO algorithm has better performance. But in comparing with CPU time, the SA algorithm is better. For comparison of the proposed algorithms, Tab. 10 and Fig. 9 are drawn. In order to compare proposed algorithms and to normalize the different measures scales, we used Norm approach [49]. Tab. 10 is normalized by Norm method (Eq. 2).

$$a'_{ij} = \frac{a_{ij}}{\sqrt{\sum_{i=1}^{m} a_{ij}^2}} \qquad (2)$$

Where a'_{ij} is the unit less of the measure of a_{ij} in jth measure of the ith algorithm.

Tab. 10. Comparison of the algorithms based on normalized value

Algorithm	SA	ACO
Total relative gap	6.559	4.478
Best solution relative gap	8.141	6.659
CPU time	0.007	0.017

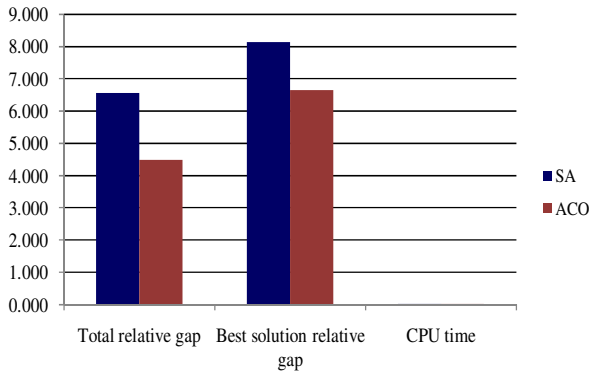


Fig. 9. Comparison of the proposed algorithms

6. Statistical Analysis

To evaluating experimental results, a statistical analysis is performed by using SPSS Statistics 17 software. An assessment of the normality of data is a prerequisite for parametric statistical tests. We normalize dataset and use well-known test of normality, namely the Shapiro-Wilk test. Then we perform an analysis of variance (ANOVA) parametric statistical test. The null hypothesis tested by one-way ANOVA is that two or more population means are equal.

The question is whether (H_0) the population means may equal for all groups and that the observed differences in sample means are due to random sampling variation, or (H_1) the observed differences between sample means are due to actual differences in the population means.

The assumptions needed for ANOVA test are: (1) random, independent sampling from the k populations, (2) normal population distributions, and (3) equal variances within the k populations. Assumption 1 is crucial for any inferential statistic. Assumptions 2 and 3 can be relaxed when large samples are used, and Assumption 3 can be relaxed when the sample sizes are roughly the same for each group even for small samples [50, 51].

The significance level is 5 percent (α=0.05). The following hypothesis is tested:

H_0: The difference between SA and ACO is not significant ($\mu_{Gap\ SA} = \mu_{Gap\ ACO}$).

H_1: The difference is significant ($\mu_{Gap\ SA} \neq \mu_{Gap\ ACO}$).

Tab. 11. The output table of ANOVA test

Source of variation	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	0.037	1	0.037	4.854	0.03
Within Groups	0.632	14	0.008		
Total	0.668	15			

Tab. 11. is the output table of SPSS Statistics software for ANOVA test. With a significance level of 5 percent, the p-value for the test is 0.03 and p-value is less than significance level (0.03 < 0.05). At the α=0.05 level of significance, there is enough evidence to conclude that there is a significant difference in the gap of the SA and ACO algorithms.

The experimental results show ACO algorithm performance is better than SA method. The results show the effectiveness of the proposed algorithms.

7. Finding the Shortest Hamiltonian Path of Iranian Cities

After solving test problems and evaluating the proposed SA and ACO algorithms, the shortest Hamiltonian path for 1071 cities is obtained. According to formal information, based on social-economic characteristics, these are the main cities of Iran. For this purpose, latitude and longitude of the cities are extracted [52]. Fig. 10 is the two dimensional view of these cities. Then, this information is converted to appropriate format for the proposed algorithms. Longitude and latitudes are converted to geographical distances on the sphere with radius 6378/388 km and are stored on a matrix with dimensional 1071 × 1071.

Tab. 12. The Hamiltonian path of the proposed algorithm for 1071 Iranian cities

Algorithm	Average of path length (km)	Average of CPU time (s)	Worst path(km)	Best path(km)
SA	32906.9	45.76	33988	32233
ACO	32755.2	205.89	33253	32039

The proposed algorithms run ten times and the averages of path length, the best and worst path (per km), and the time of each run (per seconds) are

extracted (Tab. 12). According to the number of cities, the proposed algorithms are implemented with parameters obtained for the second group problems in Tab. 3 and Tab. 5. These algorithms get distance matrix as input and output the best Hamiltonian path between 1071 cities. Fig. 11 and Fig. 12 display the best Hamiltonian path of the proposed SA and ACO algorithms, respectively. The relative superiority of the ACO algorithm to solve standard problems and to find the shortest Hamiltonian path for Iranian cities is evident. However, the SA algorithm has less CPU time than ACO algorithm.

8. Conclusions

In this paper, for the first time, two different metaheuristic algorithms, a single solution-based and a+ population solution-based algorithms, proposed to find shortest Hamiltonian path for 1071 Iranian cities, and the results are compared with each other. Standard problems were used to evaluate the performance of the proposed algorithms. To adjust the best parameter values in the proposed algorithms, DOE method was used to find the most appropriate parameters. The computational results showed slight difference in the quality of solution obtained from the proposed algorithms.

The SA algorithm solutions in standard problems are obtained in less time than the ACO algorithm, but the statistical analysis shows that solutions obtained by ACO are better than SA algorithm. In addition, the quality of solution obtained by the ACO algorithm is better than SA algorithm to find the shortest Hamiltonian path for 1071 Iranian cities, but it takes more CPU time.

For future researches, using other parameter tuning approaches such as dynamic and adaptive methods to achieve higher quality solutions and using statistical methods to evaluate the computational results of two metaheuristic algorithms are suggested.

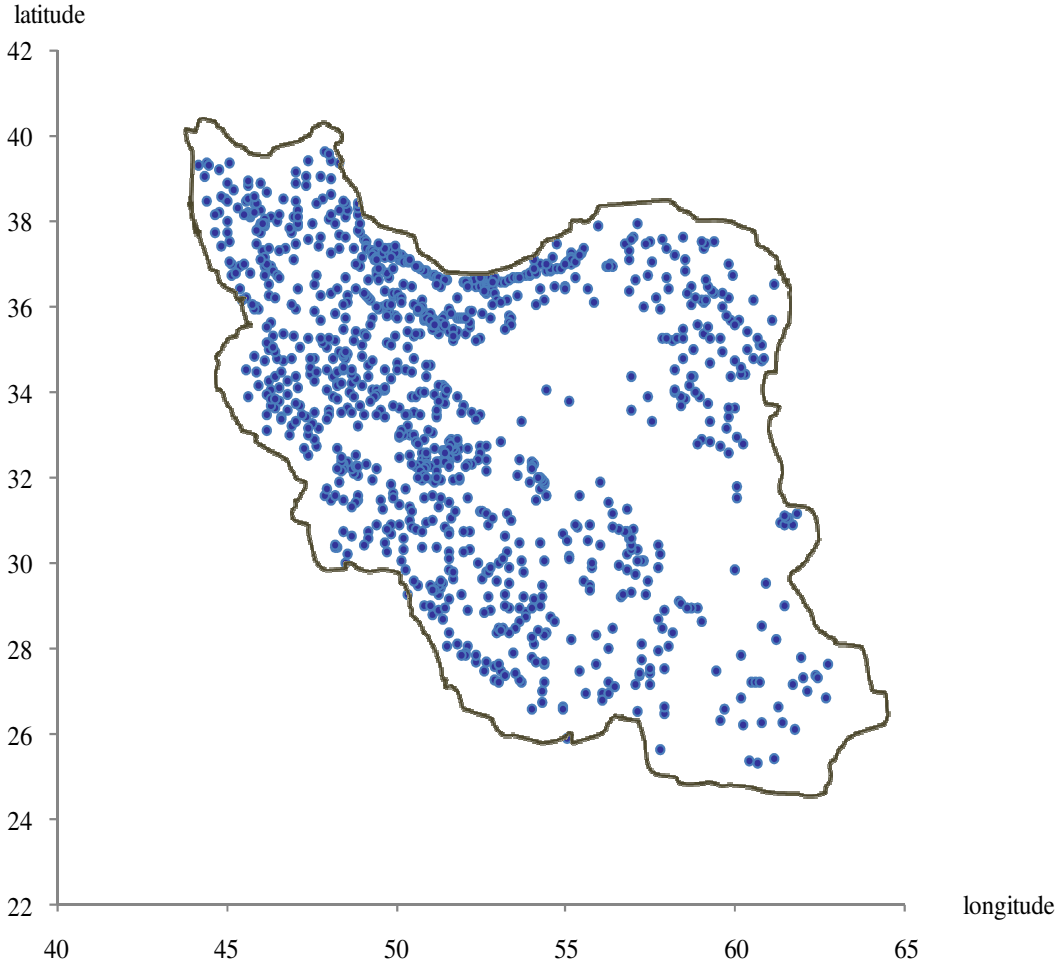


Fig. 10. Two dimensional view of 1071 Iranian cities

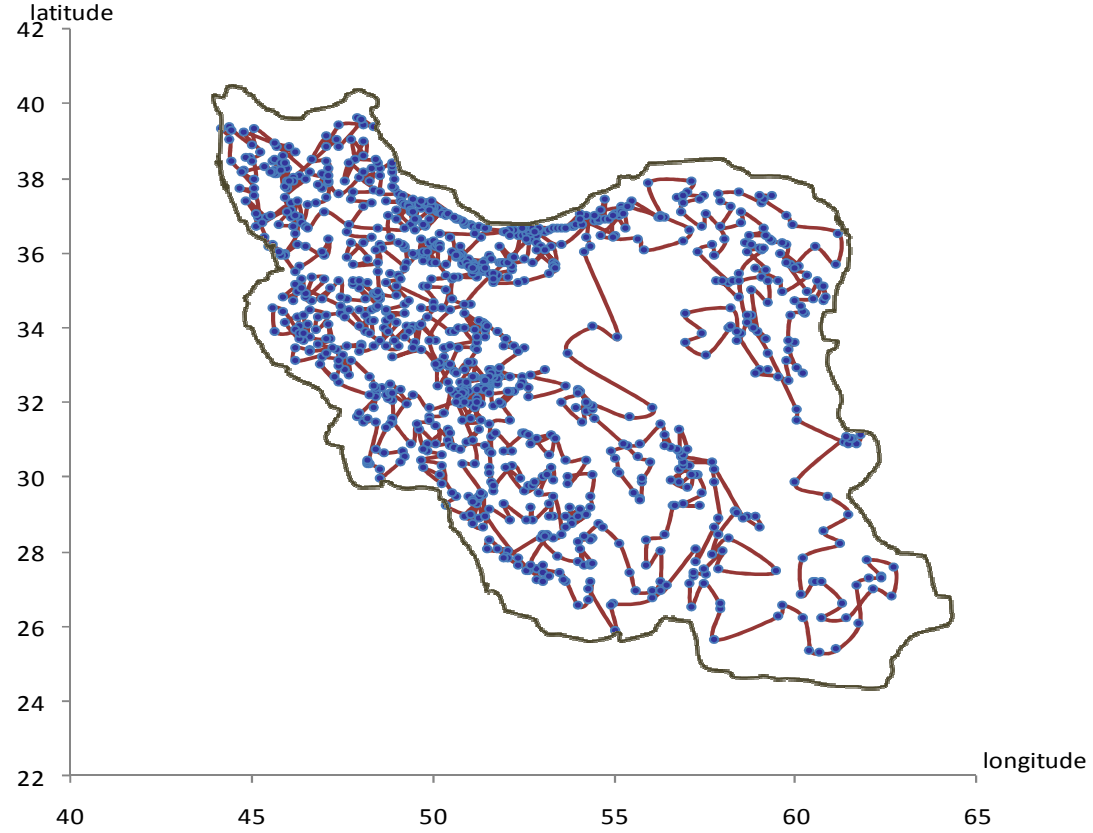


Fig. 11. The best Hamiltonian path of the SA algorithm for Iranian cities

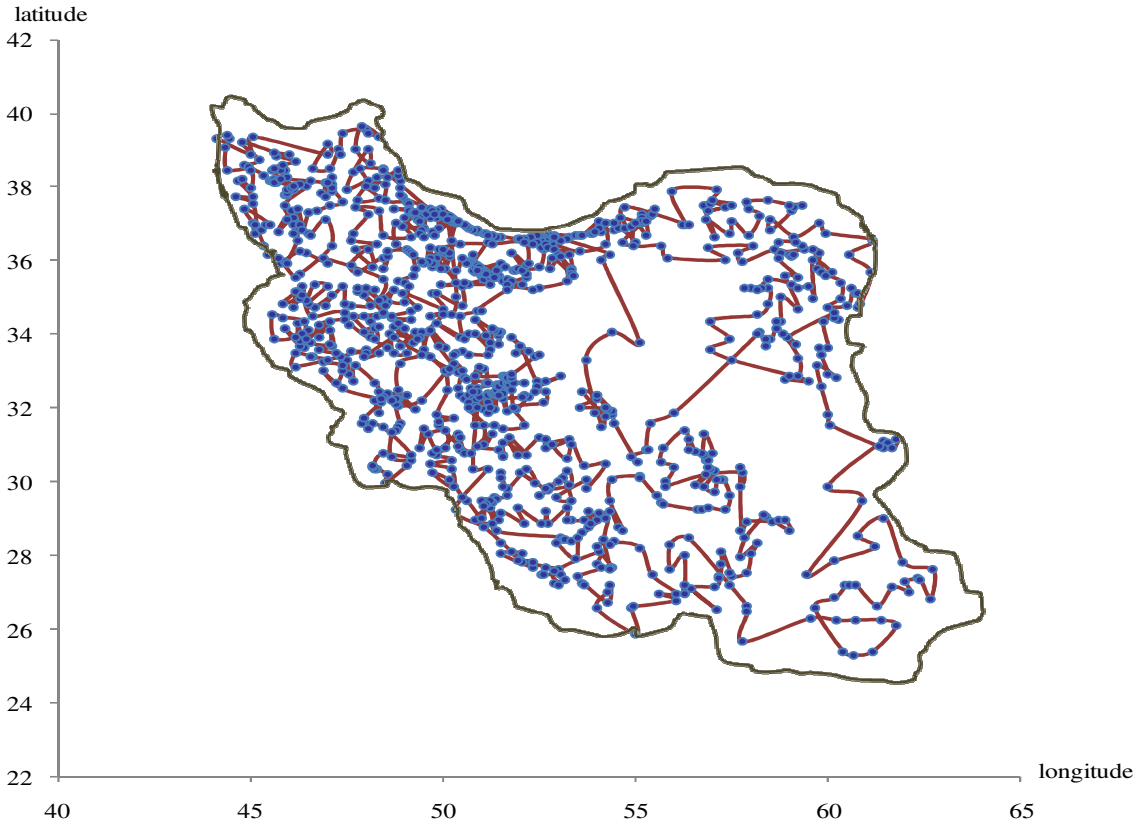


Fig. 12. The best Hamiltonian path of the ACO algorithm for Iranian cities

References

[1] Hahsler, M., Hornik, H., "*Introduction to TSP - Infrastructure for the Traveling Salesperson Problem*", Journal of Statistical Software, Vol. 23, 2009, pp. 1–21.

[2] Applegate, D., Bixby, R., Chvatal, V., Cook, V., "*On the Solution of the Traveling Salesman Problems*", Documenta Mathematica – Extra Volume Proceedings ICM III, 1998, pp. 645-656.

[3] Lawler, E., Lenstra, J., Rinnooy Kan, A., Shmoys, D., *The Traveling Salesman Problem: A Guided Tour of Cmbinatorial Optimization,* ed. Chichester: John Wiley, 1985.

[4] Johnson, S., Papadimitriou, H., *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*, In Lawler, E. L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. (ed.), Wiley, 1985.

[5] Johnson, S., Lyle, A., *The Traveling Salesman Problem: A Case Study in Local Optimization*, Local search in combinatorial optimization, In Aarts, E.H.L., Lenstra, J.K (ed.), London: John Wiley and Sons. pp. 215–310, 1997.

[6] Jaillet, P., "*Apriori Solution of a Travelling Salesman Problem in Which a Random Subset of the Customers are Visited*", Operations Research, Vol. 36, 1988, pp. 929–936.

[7] Bertsimas, D., "*Probabilistic Combinatorial Optimization Problems*", Massachusetts Institute of Technology, PhD thesis, Massachusetts Institute of Technology, Cambridge, 1988.

[8] Savelsbergh, M., "*Local Search in Routing Problems with Time Windows*", Annals of Operations Research, Vol. 4, 1985, pp. 285–305.

[9] Balas, E., "*Disjunctive Programming",* Annals of Discrete Mathematics, Vol. 5, 1979, pp. 3–51.

[10] Balas, E., "*Disjunctive Programming and a Hierarchy of Relaxations for Discrete Optimizations Problems*", SIAM Journal on Algebraic and Discrete Methods, Vol. 6, 1985, 446–486.

[11] Sarubbi, J., Miranda, G., Luna, H., Mateus, G., "*A Branch and-Cut Algorithm for the Multi Commodity Traveling Salesman Problem*", International Conference on Service Operations and Logistics, and Informatics, Beijing, 2008.

[12] Gutin, G., *Traveling Salesman and Related Problems of Graph Theory,* In Gross, J., Yellen, J. (ed.), Handbook of Graph Theory, CRC Press, Boca Raton., 2003.

[13] Noon, C., Bean, J., *"A Lagrangian Based Approach for the Asymmetric Generalized Traveling Salesman Problem"*, Operations Research, Vol. 39, 1991, pp. 623–632.

[14] Hadjicharalambous, G., Pop, P., Pyrga, E., Tsaggouris, G., Zaroliagis, C., The railway traveling salesman problem, Springer Heidelberg, 2007.

[15] Byung, I., Shim, J., Zhang, Z., "*Comparison of TSP Algorithms*", Facilities Planning and Materials Handling, 1998.

[16] Nilsson, C., "*Heuristics for the Traveling Salesman Problem*", Theoretical Computer Science Reports, Linkoping University, 2003.

[17] Dantzig, B., Fulkerson, R., Johnson, M., "*Solution of a Large-Scale Traveling Salesman Problem*". Operations Research, Vol. 2, 1954, pp. 393–341.

[18] Leonardo, Z., "*The traveling salesman problem: A comprehensive survey*", Project for CSE, 2006.

[19] Radharamanan, R. & Choi, L., "*A branch and bound Algorithm for the Travelling Salesman and the Transportation Routing Problems*", computers & Industrial Engineering, Vol. 11, 1986, pp. 236–240.

[20] Pop, P., Christos, D., Zaroliagis, Hadjicharalambous, G. "*A Cutting Plane Approach to Solve the Railway Travelling Salesman Problem*", Mathematica, Vol. 3, 2008, pp. 63–73.

[21] Zamani, R., Lau, K., "*Embedding Learning Capability in Lagrangean Relaxation: An Application to the Travelling Salesman Problem*", European Journal of Operational Research, Vol. 201, 2010, pp. 81–88.

[22] Cheng, R., Mitsuo, G., "*Crossover on Intensive Search and Traveling Salesman Problem*", Computers & Industrial Engineering, Vol. 27, 1994, pp. 485–488.

[23] Moon, C., Kim, J., Choi, G., Seo, Y., "*An efficient Genetic Algorithm for the Traveling Salesman Problem with Precedence Constraints*", European Journal of Operational Research, Vol. 140, 2002, pp. 606–617.

[24] Moscato, P., Cotta, C., "*A Gentle Introduction to Memetic Algorithms*", Operations Research & Management Science, Vol. 57, 2005, pp.105–144.

[25] Bontoux, B., Artigues, C., Feillet, D., "*A Memetic Algorithm with a Large Neighborhood Crossover Operator for the Generalized Traveling Salesman Problem*", Computers & Operations Research, Vol. 37, 2009, pp. 1844–1852.

[26] Balaprakash, P., Birattari, M., Stutzle, T., Dorigo, M., "*Estimation-Based Metaheuristics for the Probabilistic Traveling Salesman Problem*", European Journal of Operational Research, Vol. 37, 2009, pp. 1939–1951.

[27] Liu, Y., "*Different Initial Solution Generators in Genetic Algorithms for Solving the Probabilistic Traveling Salesman Problem*", Applied Mathematics and Computation, Vol. 216, 2010, pp. 125–137.

[28] Marinakis, Y., Marinaki, M., "*A hybrid multi-swarm Particle Swarm Optimization Algorithm for the Probabilistic Traveling Salesman Problem*", Computers & Operations Research, Vol. 37, 2010, pp. 432–442.

[29] Li, L. & Ju, S., "*Improved ant Colony Optimization for the Traveling Salesman Problem*", International Conference on Intelligent Computation Technology and Automation, 2008.

[30] Tiankun, L., Chen, W., Zheng, X., Zhang, Z., "*An Improvement of the Ant Colony Optimization Algorithm for Solving Traveling Salesman Problem (TSP)*", Proceedings of the 5th International Conference on Wireless communications, networking and mobile computing, Beijing, 2009.

[31] Zhang, J., "*Natural Computation for the Traveling Salesman Problem*", Second International Conference on Intelligent Computation Technology and Automation, 366–369, 2009.

[32] Ghoseari, k., Sarhadi, H., "*Finding the Shortest Iranian Hamiltonian Tour by Using Combination of ant Colony Optimization and Local Research*", In Persian, Transportation research, Vol. 2, 2009, pp. 149–161.

[33] Kirkpatrick, S., Gelatt, C., Vecchi, M., "*Optimization by Simulated Annealing*", Science, Vol. 220, 1983, pp. 671–680.

[34] Basu, S., Ghosh, D., "*A Review of the Tabu Search Literature on Traveling Salesman Problems*", IIMA Working Papers, Indian Institute of Management Ahmedabad, 2008.

[35] Jayaswal, S., "*A Comparative Study of Tabu Search and Simulated Annealing for Traveling Salesman Problem*", Department of Management Sciences, University of Waterloo, 2008.

[36] Janaki Ram, D., Sreenivas, T., Ganapathy, K., "*Functional and Parametric Tests of Integrated Circuits*", Journal of Parallel and Distributed Computing, Vol. 37, 1996, pp. 207–212.

[37] Dorigo, M., *"Optimization, Learning and Natural Algorithms"*, In Italian, PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, 1992, Milan.

[38] Dorigo, M., Maniezzo, V., Colorni, A., "*Ant System: Optimization by a Colony of Cooperating Agents*", IEEE Transactions on Systems, Man, and Cybernetic, Vol. 26, 1996, pp. 29–41.

[39] Dorigo, M., Gambardella, L. M., "*Ant Colonies for the Traveling Salesman Problem*". BioSystems, Vol. 43, 1997a, pp. 73–81.

[40] Dorigo, M., Gambardella, L.M., "*Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem*", IEEE Transactions on Evolutionary Computation, Vol. 1, 1997b, pp. 53–66.

[41] Vaughn, N., Polnaszek, C., Smith, B., Helseth, T., "*Design-Expert 6 User's Guide*", Stat-Ease, Inc, 2000.

[42] Birattari, M., Stuetzle, T., Paquete, L., & Varrentrapp, K., "*A Racing Algorithm for Configuring Metaheuristics*", In: W.B. Langdon et al. (eds.), GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference (pp. 11–18). San Francisco: Morgan Kaufmann Publishers, 2002.

[43] Montgomery, D., Design and Analysis of Experiments, 6 ed. John Wiley and Sons Inc, 2005.

[44] Adenso, B., Laguna, M., "*Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search*", Operations Research, Vol. 54, 2006, pp. 99–114.

[45] Birattari, M., *The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective*. PhD thesis, Universit´e Libre de Bruxelles, Brussels, Belgium, 2004.

[46] Ostle, B., Statistics in Research, 2nd ed. Iowa State University Press, 1963.

[47] Ridge, E., "*Design of Experiments for the Tuning of Optimization Algorithms*", PhD thesis, Department of Computer Science, University of York, United Kingdom, 2007.

[48] Reinelt, G., *TSPLIB: a Traveling Salesman Problem Library*, ORSA Journal on Computing, Vol. 3, 1991, pp.376–384.

[49] Momeni, M., New topics in operation research, 2006, Volume II, Tehran university publication.

[50] Montgomery, D.C., Runger, G., *Applied Statistics and Probability for Engineering,* 3rd ed., John Wiley, 2006.

[51] Freund, J.E., Mathematical statistics, 5th ed., Prentice-Hall, Inc, 1992.

[52] Iran Roads Map, publisher: Gitashenasi, 2008.

