Solving the Multiple Traveling Salesman Problem by a Novel Meta-heuristic Algorithm

Hossein Larki[a,*], Majid Yousefikhoshbakht[c]

[a] *Phd Student, Young Researchers & Elite Club, Hamedan Branch, Islamic Azad University, Hamedan, Iran*

[c] *Assistant Professor, Department of Mathematics, Bu-Ali Sina University, Hamedan, Iran*

Received 04 September, 2013; Revised 01 December, 2013; Accepted 23 February, 2014

Abstract

The multiple traveling salesman problem (MTSP) is a generalization of the famous traveling salesman problem (TSP), where more than one salesman is used in the solution. Although the MTSP is a typical kind of computationally complex combinatorial optimization problem, it can be extended to a wide variety of routing problems. This paper presents an efficient and evolutionary optimization algorithm which has been developed through combining Modified Imperialist Competitive Algorithm and Lin-Kernigan Algorithm (MICA) in order to solve the MTSP. In the proposed algorithm, an absorption function and several local search algorithms as a revolution operator are used. The performance of our algorithm was tested on several MTSP benchmark problems and the results confirmed that the MICA performs well and is quite competitive with other meta-heuristic algorithms.

Keywords: Imperialist Competitive Algorithm, Multiple Traveling Salesman Problem, Lin-Kernigan Algorithm, NP-hard Problems.

1. Introduction

The multiple traveling salesman problem (MTSP) is a generalization of the well-known traveling salesman problem (TSP)(Ahmadvand et al., 2012) which is an obviously NP-hard problem (Evelyn et al., 2007). The MTSP is more difficult than TSP, because it involves finding a set of Hamilton circuits without sub-tours for $m>1$ salesmen to serve a set of $n>m$ nodes so that each one is visited by exactly one salesman. The objective is to minimize the total distance traveled by the salesmen. Many further requirements and operational constraints are imposed on the route construction in practical applications of the MTSP. For example, the variants of MTSP include: pickup and delivery (If the salesmen need to pick up loads) (Wang and Regan, 2002), time windows (If the services have time constraint) (Kim and Park, 2004), multi-depot (if the problem has a single depot or multiple depots) (Rathinam and Sengupta, 2006), open ends (if the salesmen have not returned to the depot) (Sariel and Balch, 2005) and others. Moreover, the MTSP can be used to model many practical problems such as the hot rolling planning (Huang and Wang, 2007) newspaper production/distribution (Carter and Ragsdale, 2002), Print press scheduling (Gorenstein, 1970), Crew scheduling (Bektas, 2006), Interview scheduling (Gilbert and Hofstra, 1992), Mission planning (Brummit and Stentz, 1998), Hot rolling scheduling (Tang et al., 2000), Design of global navigation satellite system surveying Networks (Saleh and Chelouah, 2004), etc.

In all the problems, scheduling and vehicle scheduling are the most commonly researched areas. The scheduling of jobs on a production line is often modeled as a TSP. If the production operation is expanded to have multiple parallel lines to which the jobs can be assigned, the problem can be modeled as a MTSP (Sedighpour et al., 2011). Another problem that is often modeled as a MTSP is the vehicle scheduling problem (VSP). The VSP consists of scheduling a set of vehicles which leave from, and return to a common depot to visit a number of locations so that each location is visited exactly once (Park, 2001). Moreover, if the capacity restrictions are removed, then the MTSP can be considered as a relaxation of the vehicle routing problem (VRP). In other words, all the formulations and solution approaches proposed for the VRP are also valid and applicable to the MTSP by assigning sufficiently large capacities to the salesmen (vehicles). Moreover, when there is only a single salesman, the MTSP reduces to the well-known TSP (Oberlin et al., 2009; Gutin and Punnen, 2002).

There have been important advances in the development of exact, heuristic and meta-heuristic algorithms for solving the MTSP. Although, exact algorithms are appropriate in instances with small sizes;

* Corresponding author Email address: h.larki@aut.ac.ir

they are not often suitable for real instances owing to the computational time required to obtain an optimal solution. There are several exact algorithms which were used to solve the MTSP such as Cutting Planes algorithm (Laporte and Nobert, 1980), Branch-and-Bound algorithm (Saad et al., 2013), and Lagrangian algorithm (Yadlapalli et al., 2009).

Because of the fact TSP belongs to the class of NP-hard problems (Russell, 1977), it is obvious that MTSP is an NP-hard problem. This means that the MTSP solution time grows exponentially with the increase in distribution points thus exact algorithms are not capable of solving problems with large dimensions. Therefore, heuristics are thought to be more efficient for complex MTSP and have become very popular among researchers. A number of heuristic algorithms have been performed on the MTSP, such as *k*-opt approach (Potvin, 1989) and minimum spanning tree (Malik et al., 2007).Since the computational time required to solve adequately large problem instances is still prohibitive and heuristic methods become trapped in local optima and cannot gain a good suboptimal solution for solving NP-hard problems, the focus of most researchers has been toward the design of meta-heuristic approaches capable of producing high quality near optimum solutions within a reasonable computational time. As a result, many recent studies have been published on using advanced meta-heuristic techniques to solve MTSP. Some of the well-known meta-heuristics which have more capability to find optimal solution are the genetic algorithm (Zhou and Li, 2010), evolutionary algorithm (Sofge et al., 2002), neural networks (Modares et al., 1999), tabu search (Ryan et al., 1998) and ant colony optimization (YousefiKhoshbakht et al., 2013).

One of the approaches used for solving the MTSP is to transform the problem into a standard TSPin order to use TSP algorithms for solving the MTSP. In this method, ***m*** salesmen and ***n*** nodes in a MTSP problem are transformed into ***n+m+1*** nodes in TSP,in which ***m+1*** nodes from among ***n+1*** nodes to ***n+m+1*** nodes from the artificial depots. However, the resulting TSP has highly degenerated (Bektas, 2006) when an MTSP is transformed into a single TSP; since the resulting problem is more arduous to solve than an ordinary TSP with the same number of nodes. Some of the well- known transformation algorithms are branch-and-bound algorithm (Laporte et al., 1988) and sub tour elimination algorithm (Orloff, 1974).

Among the prominent problems in the distribution and logistics,TSP and its extensions have a central role. Although TSP has been studied for many years due to its practical applications in real world logistics and transportation problems, MTSP has not been considered as an attractive problem and researchers have failed to pay attention to this problem. Therefore, a new meta-heuristic algorithm based on Modified Imperialist Competitive Algorithm and Lin-Kernigan algorithm (MICA) is proposed for solving the MTSP in this paper. In more details, several local search algorithms as a revolution operator and an absorption function are applied in order to improve the algorithm. Furthermore, the Lin-Kernigan algorithm as one of the most powerful algorithms is applied when a new solution is obtained during the MICA algorithm. In the following parts of this paper, in Section 2, the Imperialist Competitive Algorithm (ICA) and Lin-Kernigan approach are explained in more detail and then, the combination of these two algorithms are also described. In section 3, the efficiency and the performance of the MICA are elaborated and compared with some of the other algorithms on MTSP problems. Finally, the conclusions are presented in section 4.

2. Our Approach

In this section, first the ICA is presented and then the Lin-Kernigan algorithm is explained. Finally, the hybrid algorithm will be analyzed in more detail.

2.1. Imperialist Competitive Algorithm

During the last decade, nature-inspired intelligence became increasingly popular through the development and utilization of intelligent paradigms in solving combinatorial optimization problems. Among the most popular nature-inspired approaches, those methods representing successful animal and micro-organism team behavior, such as swarm or flocking intelligence, artificial immune systems, optimized performance of bees and ant colonies to ant colony optimization, etc. have been considered by researchers in recent years.

One of the new social-politically motivated global search strategies, which proved to have an acceptable performance for some of benchmark cost functions, is the ICA. This method introduced by AtashpazGargari et al. in 2007 (Atashpaz-Gargari and Lucas, 2007) uses socio-political evolution of human as a source of inspiration for developing a powerful optimization strategy. This algorithm considers the imperialism as a level of human social evolution by mathematically modeling this complicated political and historical process.

The ICA has been applied successfully in different domains namely designing controllers (Rajabioun et al., 2008), recommender systems, characterization of Elasto-plastic properties of materials (Biabangard-Oskouyi et al., 2008), traveling salesman problem (YousefiKhoshbakht and Sedighpour, 2013), and many other optimization problems (Khabbazi et al., 2009). The algorithm is a novel global search strategy that uses imperialism and imperialistic competition process as a source of inspiration. The ICA is based on the fact that countries try to extend their power over the other countries in the real world for using their resources and bolstering their own government. The pseudo-code of the ICA is presented in Figure 1, while in the current section the procedures of the algorithm are explained in detail.

```
Step 1: Initialize countries including colonies and imperialists.
Step 2: Move the colonies toward their relevant imperialist.
Step 3: Revolvep percent of colonies.
Step 4: If there is a colony in an empire which has better cost than the imperialist, exchange the positions of
        that colony and its imperialist.
Step 5: Compute the total cost of all empires.
Step 6: Pick the weakest colony from the weakest empires and give it to the best empire.
Step 7: If there is an empire with no colonies, eliminate that empire.
Step 8: Save the best so far solutions.
Step 9: If stop conditions are not satisfied, go to step 2.
```

Fig.1. Pseudo-code of the ICA

In ICA, the first step is to generate an initial population like other evolutionary algorithms. The population set includes a number of feasible solutions called a ‘country’, which corresponds to the term ‘chromosome’ in the GA method. These countries are of two types: colonies and imperialists; that all together form some empires. Bigger and stronger empires have more colonies than smaller and weaker ones.After forming initial empires, their colonies start moving toward their relevant imperialist country. This movement is a simple model of assimilation policy which was pursued by some of the imperialist states.

If after this movement one of the colonies possesses more power than its relevant imperialist, they will exchange their positions. To begin the competition between empires, the total objective function of each empire should be calculated. It depends on the objective function of both an imperialist and its colonies. Imperialistic competition among these empires forms the basis of the proposed evolutionary algorithm. During this competition, weaker empires which have lost all their colonies collapse; and powerful ones take the possession of their colonies. At last, the most powerful empire will take the possession of other empires and wins the competition. In other words, imperialistic competition hopefully converges to a state in which there exists only one empire and its colonies are in the same position and have the same cost as the imperialist.

2.2. Lin-Kernigan algorithm

In order to help ICA during the algorithm steps and to prevent any stagnation, a local searching algorithm is used when a better solution compared to previous iterations is attained. In fact, the probability of finding better solutions near a good solution is relatively high. There exist many algorithms such as 2-opt algorithm (Sedighpour et al., 2013) for the local search and they of course have pros and cons. This algorithm starts with a feasible tour and continues by omitting two arcs of the tour which are not adjacent and connecting them again by another method in such a way that the new tour length is shorter. Omitting two arcs and again connecting them are repeated until no improving 2-opt is found. Here the Lin-Kernigan algorithm is applied as a simple algorithm. Besides, the 2-opt local search is a unique case of the ω-opt algorithm, where in each step ω links of the current tour are replaced by ω links in such a way that a shorter tour is achieved. The time complexity for testing ω–exchange is equal to $O(n^{\omega})$ in which there is no nontrivial upper bound of the number of ω–exchanges.

The Lin-Kernighan algorithm which is a generalization of this simple principle form removed this disadvantage by introducing a great variable ω-opt algorithm in which the value of ω is changed during its execution. At each iteration the algorithm considers a growing set of potential exchanges which start with $\boldsymbol{r} = 2$. These exchanges are chosen in such a way that a feasible tour may be formed at any stage of the process. Then, the problem is examined for ascending values of ω and the algorithm decides what the value of ω should be. If an interchange of λ links succeeds in finding a new shorter tour, then the actual tour is replaced with the new tour. This continues until some stopping conditions are satisfied.

2.3. Hybrid Algorithm

In the MICA, only one kind of feasible solutions or primitive countries is commonly employed for solving the MTSP. Therefore, a bisection array shown in Figure 2 is used (where $\boldsymbol{n} = 11$ and $\boldsymbol{m} = 3$). In this array, the visited nodes are ordered from left to right in the first section and the number visited nodes by each salesman are shown in the second section. In this technique, the $\boldsymbol{n}$ nodes are represented by a permutation of the integers from 1 to $\boldsymbol{n}$. This permutation is partitioned into $\boldsymbol{m}$ sub-tours by the insertion of $\boldsymbol{m}$ positive integers (from 1 to $\boldsymbol{m}$) that represent the change from one salesman to the next. In the example in Figure 2, the first salesman visits nodes 1 and 9 (in that order), the second salesman visits nodes 10, 3, 11, 5, 4 and 2 (in that order), and the third salesman visits nodes 6, 7 and 8 (in that order).

Therefore, a defined and variable number $\boldsymbol{p}$ of primitive solutions must be randomly generated by the user and the values of the objective function f_i for each $i = 1,...,p$ must be obtained. It is noted that using a random construction at this level leads to obtaining solutions that have an irregular construction in a feasible space. Then, ***e*** countries that have better objective function are selected and called empire countries. Furthermore, the number of colonies devoted to each empire country is equal. Therefore, each empire increases its quality using the imperialist countries which play local optima role. What is worthy of note is the fact that, since a lot of possible points are combined with local optima, an absorption function must be used which includes randomization concept so that the results of combinations will not yield a very similar response.

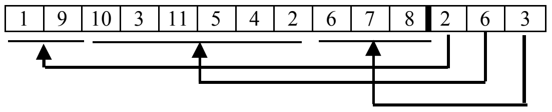


Fig. 2.A Country in ICA

One of the best absorption functions in terms of quality and speed is still Order crossover in genetic algorithm. For that reason, this method- which is simple to implement-has been considered here and the modified absorption function is proposed based on the Order crossover. In this absorption function, a random number between 2 to ***n-1*** is selected. After that, some nodes equal to the selected number are chosen from the colonized countries and are arranged according to the order of the imperialist countries. For instance, in Figure 3, if the selected number is 3, then 3 nodes like 3, 6, and 5 are chosen from the colonies. Then, their order in imperialist country which is 3, 5 and 6 in the example is found out. Therefore, the new result for the colony will be [2 3 1 5 6 4| 2 4]. Clearly this method allows only the generation of valid strings.

The absorption function is performed for all colonies in comparison to their imperialist countries, and the results and the values are replaced with the colonies' results and values proved that the new results are better. At the next stage, ***p*** percent of countries experience revolution. This causes variety of colonies in the empire, and if possible, their quality increases at each stage. The proposed method for this stage is the 0–1 and 1–1 Exchange moves. In the 0–1 Exchange move, a candidate node is removed from its origin route and inserted in the best position. However, in 1–1 Exchange a node is swapped with another node. These replacements are done if the new results are better than the previous ones.

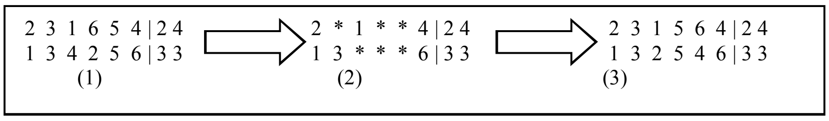


Fig.3. Absorption function

After the results are calculated for all colonies, these countries might have a better objective function compared to their respective imperialists. Therefore, a colony with the best value in each empire is chosen and if it possesses a better objective function it replaces an imperialist country. In case there are a few colonies with the same objectivefunction, one of them is chosen randomly and is compared with an imperialist country in the empire. From this stage up to the end of the algorithm, there will not be any changes in objective functions of feasible values. Therefore, the best results and values of the objective function must be saved. For this purpose, two variables are chosen in order to save the best results and values until the current iteration. In each iteration, after the imperialist countries were replaced, the best results for the imperialist countries are chosen as the best current results. If the new attained value is better than the value of the previous iteration, local search is conducted and the previous results and values are replaced with the new results and values. Up to this stage in the algorithm, the purpose is conducting a general search to locate important areas for algorithm convergence. Now, important areas must be identified and the population must be converged toward them. After this stage, some of the initial population moves toward these areas. The power of empires is assessed at this stage. In imperialist competitions, more powerful empires must expand their territory through occupying other countries. In order to achieve this goal, the power of the empire is calculated using formula (1).

$$h_j = f_j + \lambda(s_j) \qquad j = 1,...,e \tag{1}$$

In this formula h_j, s_j, and λ represent empire's total power, the average objective function of the colonies in each empire, and the [0 1] impact coefficient which determines the relative power of a colony compared to an empire, respectively. A weaker empire loses its power by losing its weakest colony to the strongest empire. Moreover, in order to help the MICA so that it does not lead to stagnation, a local searching algorithm is used when the algorithm attained a better solution compared to previous iterations. In fact, the probability of finding better solutions near a good solution is relatively high. There exist many algorithms for the local search and they of course have pros and cons. Here the Lin-Kernigan algorithm is used as one of the most simple and successful methods for generating optimal or near optimal solutions for the TSP. At this stage, the final condition is checked and if it is met, the algorithm ends. Otherwise, the algorithm is iterated by returning to absorptionfunction step.

To end the loop, one of the two conditions must be met: the iteration of algorithm 2***n*** times or the survival of just one empire. If each of the conditions is met, the algorithm ends and the obtained results and values up to now are considered as the best values and results of the algorithm.

3. Computational Experiments

In this section, some numerical results of the comparison between the MICA and some meta-heuristic algorithms are presented. To reveal the variability of the MICA's performance from one run to another, 10 runs are carried out for each instance and the best answer was registered. The proposed algorithm was coded using MATLAB language executed on a computer with a 1.00 GB RAM and an Intel, 2.80 GHz CPU. Because changing the value of any parameter could affect the optimum value of the others, finding the best values for the parameters of the MICA is a complicated problem itself. What is considered in this paper is an experimental approximation of the best values. In our work, we adopted the values {***n/3, n/2, n, 3n/2, 2n***} for ***P*** as number of initial populations and the values {***P/6, P/5, P/4, P/3, P/2***} for ***e*** as number of imperialists. Finally, for the rest of the λ as empire's total power in formula (2) we implemented a set of experiments on the test problem generated with λ □ {0.1, 0.2, 0.3, 0.4, 0.5}.

The ranges of three parameters were set in Table 1. In this table, all of the parameter values have been determined on the MTSP-100-I instance with 10 salesmen by the numerical experiments. Then to determine the value of parameters several alternative values for each parameter were tested, while all the others were held constant, and the ones that were selected gave the best computational results concerning the quality of the solution. Although the results confirmed that our parameter setting worked well, it is also possible that better solutions could exist.

Table 1
Parameter setting for the MICA

Parameters	Candidate Value	The best value
P	n/3, n/2, n, 3n/2, 2n	2n
E	P/6, P/5, P/4, P/3, P/2	P/5
λ	0.1, 0.2, 0.3, 0.4, 0.5	0.3

Since there were no standard existing benchmarks for MTSP, this study generated some small and large test problems. All the examples are randomly located over a square with no route length restrictions. Therefore, a new set consisting of five small tests numbered from ***A1*** to ***A5***with sizes ranging from 20 to 60 nodes including the depot were considered. Because the commercial linear programming software including ILOG and Cplex could find optimal solutions for the small-scale of the problems, like MTSP, hence it can be used to evaluate the accuracy of the proposed model. Therefore, we show characteristics of instances and the results obtained by AIMMS and the proposed algorithm on the set of benchmark instances in Table 2. The information in this table is consisting of the number of nodes, the number of salesmen, the solution costs obtained from AIMMS, the running time in seconds of AIMMS, the best solution costs of MICA and its computing time. Finally in the last column, the percentage difference (Gap) of the MICA solution cost comparison to the AIMMS solution costs was shown. The Gap is calculated by below formula.

$$\text{Gap}=100\times\frac{\left(\text{value of MICA }-\text{ Value of AIMMS}\right)}{\text{value of AIMMS}}$$

Table 2
Comparison results for problem set

Instance	n	m	**AIMMS Cost**	Time (Sec)	**MICA Cost**	Time (Sec)	Gap
A1	20	5	**210.45**	15	**210.45**	8.94	0
A2	30	7	**357.51**	430	**357.51**	12.97	0
A3	40	8	**1259.84**	1652	1274.14	21.84	-1.12
A4	50	8	845.21	12473	**831.87**	18.45	1.60
A5	60	9	798.54	29457	**759.64**	20.54	5.12

Based on this table, AIMMS obtained the optimal solution only for instances A1, A2 and A3 and the other instances automatically terminated before reaching to optimal solution. The results show that the MICA algorithm produced the optimal solutions for two out of the five problem instances in a reasonable time. Furthermore, this algorithm can obtain the better solutions than AIMMS in A4 and A5 and obtain equal solutions with AIMMS for A1 and A2. Only for instances A3, the AIMMS can gain better feasible solution against MICA. Finally, the proposed algorithm in average improves the solution cost as much as 1.12% of these instances compared to AIMMS.

The experimental 21 instances including MTSP-51, MTSP-100-I, MTSP-100-II, MTSP-150-I and MTSP-150-II with different salesmen (***m***) and problem sizes (***n***) combinations are summarized in Table 3. This set including depot has 51, 100, 150 nodes. Consequently, the instances which are Euclidean, two-dimensional symmetric problems (Brown et al., 2007; Carter and Ragsdale, 2006) are utilized to test our algorithm. Besides, the best, worst, average and time of solutions of the MICA are also presented in this table.

In Table 4, the average results obtained from calculations by the MICA are compared with other algorithms including grouping genetic algorithm (GGA) (Brown et al., 2007), new grouping genetic algorithm (NGGA) (Singh and Baghel, 2009), genetic algorithm (GA) (Carter and Ragsdale, 2006) and sweep algorithm, elite ant system (SA+EAS) (Yousefikhoshbakht and Sedighpour, 2012) and ICA (Yousefikhoshbakht and Hashemi, 2013).

Moreover, in order to show the method's performance more clearly, we present the best known solutions (BKS) that have been published in the related literature in this table. It should be noted that a new group GA algorithm was proposed for these instances (in Singh and Baghel, 2009) and the results were observably improved compared with GA and GGA.

Table 3
Benchmark instances

instance	Number	m	n	Best	Worst	Average	Time
MTSP-51	1	3	51	447	447	447	2.12
	2	5	51	473	477	474	3.45
	3	10	51	584	584	584	9.51
MTSP-100-I	4	3	100	22127	22223	22173	6.45
	5	5	100	23398	23739	23531	12.89
	6	10	100	27755	28121	27921	24.41
	7	20	100	40212	40625	40345	45.78
MTSP-100-II	8	3	100	22051	22245	22112	4.99
	9	5	100	23678	23819	23741	12.02
	10	10	100	27724	27982	27895	29.81
	11	20	100	39822	38899	39842	67.38
MTSP-150-I	12	3	150	6632	6741	6692	9.83
	13	5	150	6751	6831	6789	21.54
	14	10	150	7672	77722	7693	35.37
	15	20	150	9831	9988	9901	97.24
	16	30	150	13652	12978	13851	121.67
MTSP-150-II	17	3	150	38499	38721	38553	13.62
	18	5	150	39962	40361	40123	22.56
	19	10	150	44191	44321	44221	48.02
	20	20	150	55670	55891	55762	76.41
	21	30	150	71101	71870	71431	118.94

The results of this comparison show that the proposed algorithm gains better solutions than the GGA and NGGA in all of the 21 instances. Moreover, computational results of the MICA and GA show that these algorithms have a stiff competition and the proposed algorithm produces 12 better solutions than GA including 1, 2, 4, 5, 6, 7, 10, 12, 14, 15, 16, 19, 20 and 21 which are not small scale problems. Moreover, the proposed algorithm yields worse solutions than the GA in 8, 9, 11, 12, 13, 17 and 18. In other words, the performance of the MICA is better in reaching the sub-optimal solution than the GA. The results also indicate that although the SA+EAS provide a better solution than the proposed algorithm does for 5, 6, 10, 12 and 21, this algorithm cannot maintain this advantage in other examples. Finally, the proposed algorithm has a near competition with the ICA in which the MICA has a 12 better solutions than ICA despite the ICA obtain 7 solutions with more quality than the proposed algorithm. Therefore, the computational experiments confirm that in general the proposed algorithm provides better results compared to the GA in three literatures, SA+EAS and ICA in terms of the quality of the solutions.

Table 4
Comparison of algorithms for standard problems of MTSP

instance	Number	GGA	NGGA	GA	SA+EAS	ICA	MICA	BKS
MTSP-51	1	924	543	449	447	447	447	447
	2	882	586	479	475	479	474	475
	3	1001	723	584	586	584	584	584
MTSP-100-I	4	79347	-	22100	22254	22100	22173	22100
	5	70871	-	23398	23398	23398	23531	23398
	6	89778	-	28356	27801	27942	27921	27801
	7	137805	-	41554	40534	40688	40345	40534
MTSP-100-II	8	-	26653	22051	22123	22051	22112	22051
	9	-	30408	23678	23678	23678	23741	23678
	10	-	31227	28488	27830	27925	27895	27830
	11	-	54700	40892	39882	40985	39842	39882
MTSP-150-I	12	33888	-	6632	6632	6632	6692	6632
	13	26851	-	6751	6842	7098	6789	6751
	14	37771	-	7885	7753	7696	7693	7696
	15	43699	-	10399	10321	10399	9901	10321
	16	52564	-	14929	13989	13652	13851	13652
MTSP-150-II	17	-	47418	38434	39006	39342	38553	38434
	18	-	49947	39962	40437	39962	40123	39962
	19	-	54958	44274	44255	44652	44221	44255
	20	-	73934	56412	56001	56412	55762	56001
	21	-	99547	72783	71223	71965	71431	71223

Figure 4 shows a comparison between Gap values of the meta-heuristic algorithms, where the gap is defined as the percentage of deviation from the best known solution in the literature. In this figure, the horizontal axis shows the name of instances and the vertical axis indicates the Gap of the proposed algorithm from the BKS. Here, the Gap is equal to $100\times[c(s^{**})-c(s^{*})]/c(s^{*})$ where s^{**} is the best solution found by the algorithm for a given instance, and s^{*} is the overall best known solution for the same instance on the Web.A zero gap indicates that the best known solution is found by the algorithm. The results show that not only the MICA finds high quality solutions for all of the benchmark problems but also the new mean solutions for 6 out of 21 problems published in the literature are found by the MICA. However, in all the instances, the gap is below 1%, and for overall the average difference is -0.12%.

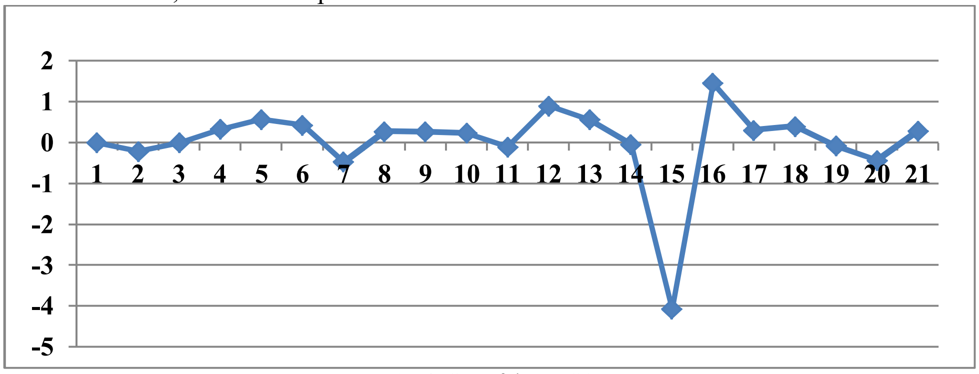
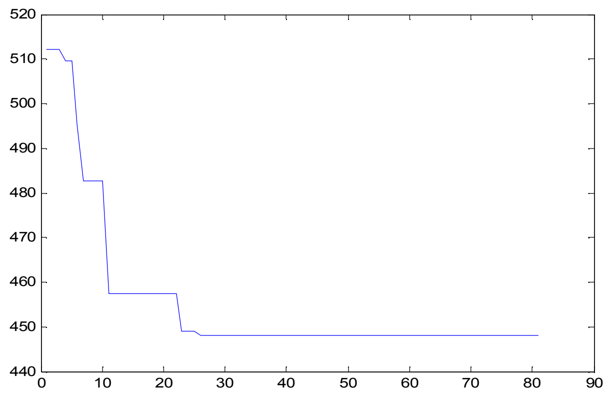


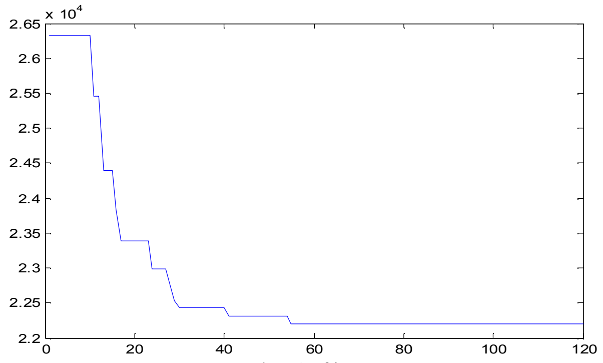
Fig. 4. Gap of the MICA

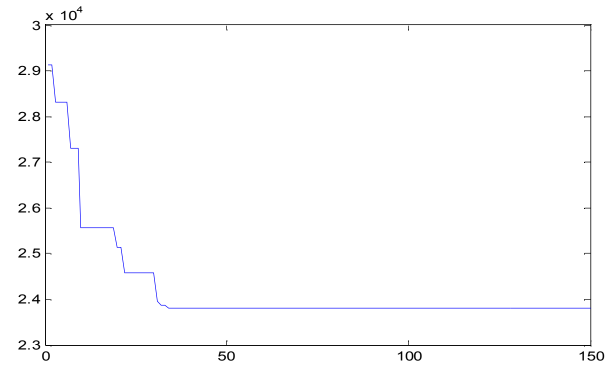
To show exactly that the proposed algorithm is effective, figure 5 shows 4 MTSP instances including 1, 4, 9, 19 from Table 1 with 50, 100 and 150 nodes. This figure represents the results and the convergence curve of the MICA. In four curves, the vertical axis shows the total distance gained by the algorithm and the horizontal axis shows the number of iterations as one of the termination conditions of the MICA.

4. Conclusion

In this paper, a hybrid algorithm called MICA was proposed for solving the MTSP. This algorithm is more efficient than ICA, SA+EAS and the modified genetic algorithms for dealing with MTSP. The algorithm was tested in 26 benchmark problems with 20–150 nodes and it was found capable of improving the BKS of 6 instances. Our results also show that the gap of the MICA stays on the average below -0.12%. Computational experiments with benchmark problem instances showed that in general the proposed MICA yeilds better results compared to the existing solution methods for MTSP in term of the solution quality. It seems that the combination of the proposed algorithm and tabu search or ant colony optimization will yield better results. Besides, using this proposed algorithm for other versions of the MTSP and also applying this method in other combinational optimization problems including the vehicle routing problem, school bus routing problem and the sequencing of jobs are suggested for future research.


Number 1 of instances


Number 4 of instances


Number 9 of instances

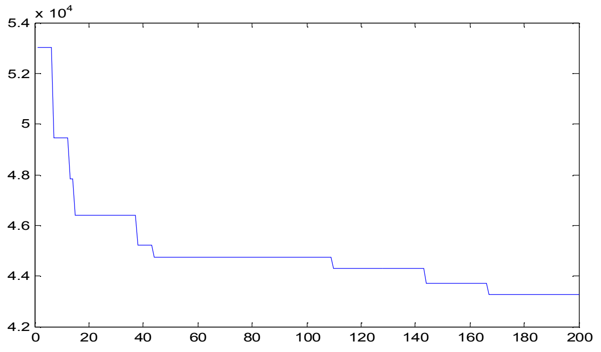

Number 19 of instances

Fig. 5. Execution plot for four instances in Table 1

5. Acknowledgement

The authors would like to acknowledge the Young Researchers & Elite Club, Hamedan Branch, Islamic Azad University for their financial support of this work.

References

[1] Ahmadvand, M., YousefiKhoshbakht, M. and MahmoodiDarani, N. (2012). Solving the Traveling Salesman Problem by an Efficient Hybrid Metaheuristic Algorithm, Journal of Advances in Computer Research, 3(3), 75-84.

[2] Atashpaz-Gargari, E. and Lucas, C. (2007). Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition, In: Proceedings of the IEEE Congress on Evolutionary Computation, Singapore.

[3] Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures, Omega, 34, 209–219.

[4] Biabangard-Oskouyi, E., Atashpaz-Gargari, N., Soltani, M. and Lucas, C. (2008). Application of Imperialist Competitive Algorithm for materials property characterization from sharp indentation test, International Journal of Engineering Simulation.

[5] Brown, E. C., Ragsdale, C. T. and Carter, A. E. (2007). A grouping genetic algorithm for the multiple traveling salesperson problem, International Journal of Information Technology & Decision Making, 6(2), 333–347.

[6] Brummit, B. and Stentz, A. (1998). GRAMMPS: a generalized mission planner for multiple mobile robots. Proceedings of the IEEE international conference on robotics and automation.

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

[8] Carter, A.E. and Ragsdale, C.T. (2002). Scheduling pre-printed newspaper advertising inserts using genetic algorithms, Omega, 30, 415–421.

[9] Gutin, G. and Punnen, A.P. (2002). Editors. Traveling salesman problem and its variations, Dordrecht: Kluwer Academic Publishers.

[10] Gilbert, K. C. and Hofstra, R. B. (1992). A new multi period multiple traveling salesman problem with heuristic and application to a scheduling problem, Decision Sciences, 23, 250–259.

[11] Gorenstein, S. (1970). Printing press scheduling for multi-edition periodicals, Management Science, 16(6), 373–383.

[12] Evelyn, C. Brown, C. T. Ragsdale, A. Carter, A. E. (2007). A grouping genetic algorithm for the multiple traveling salesperson problem, International Journal of Information Technology & Decision Making, 6(2), 333–347.

[13] Huang, K. and Wang, D. (2007). Multiple traveling salesman problem and its application to hot rolling planning, Application Research of Computers, 24 (7), 43-46.

[14] Khabbazi, A., Atashpaz, E. and Lucas, C. (2009). Imperialist competitive algorithm for minimum bit error rate beamforming, international journal of Bio-Inspired computation, 1, 125-133.

[15] Kim, K. H. and Park, Y.M. (2004). A crane scheduling method for port container terminals, European Journal of Operational Research, 156, 752–768.

[16] Laporte, G. and Nobert, Y. A. (1980). Cutting planes algorithm for the m-salesmen problem. Journal of the Operational Research Society, 31, 1017–1023.

[17] Laporte G, Nobert Y, Taillefer S. (1988). Solving a family of multi-depot vehicle routing and location-routing problems. Transportation Science 22(3), 161–172.

[18] Orloff, C.S. (1974). Routing a fleet of M vehicles to/from a central facility, Networks 4, 147–162.

[19] Malik, W., Rathinam, S. and Darbha, S. (2007). An approximation algorithm for a symmetric Generalized Multiple Depot, Multiple Travelling Salesman Problem. Operations Research Letters, 35(6), 747-753.

[20] Modares, A., Somhom, S. and Enkawa, T. (1999). A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems. International Transactions in Operational Research, 6(6), 591-606.

[21] Oberlin, P., Rathinam, S., Darbha, S. (2009). A transformation for a Multiple Depot, Multiple Traveling Salesman Problem, American Control Conference, 2636 – 2641.

[22] Park, Y.B. (2001). A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines, International Journal of Productions Economics, 73 (2), 175–188.

[23] Potvin, J., Lapalme, G. and Rousseau, J. (1989). A generalized k-opt exchange procedure for the MTSP. Information Systems and Operations Research, 27(4), 474–481.

[24] Rajabioun, R., Hashemzadeh, F., atashpaz-Gargari, E., Mesgari, B. and Salmas F. R. (2008). Decentralized PID Controller Design for a MIMO Evaporator Based on Colonial Competitive Algorithm, in 17th IFAC World congress, Seoul, Korea.

[25] Rathinam, S. and Sengupta, R. (2006). Lower and Upper Bounds for a Symmetric Multiple Depots, Multiple Travelling Salesman Problem, Research Report, UCB-ITS-RR-2006-2.

[26] Russell, R. A. (1977). An effective heuristic for the m-tour traveling salesman problem with some side conditions. Operations Research, 25(3), 517–524.

[27] Ryan, J. L., Bailey, T. G., Moore, J.T. and Carlton, W.B. (1998). Reactive Tabu search in unmanned aerial reconnaissance simulations, Proceedings of the 1998 winter simulation conference, 1, 873–879.

[28] Saad, S.,Nurhadani, W., Jaafar, W. and Jamil, S. J. (2013). Solving standard traveling salesman problem and multiple traveling salesman problem by using branch and bound AIP Conf. Proc. 1522, 1406.

[29] Saleh, H. A. and Chelouah, R. (2004). The design of the global navigation satellite system surveying networks using genetic algorithms, Engineering Applications of Artificial Intelligence, 417, 111–122.

[30] Sariel, S. and Balch, T. (2005). Real Time Auction Based Allocation of Tasks for Multi-Robot Exploration Problem in Dynamic Environments. In the 20th National Conference on Artificial Intelligence (AAAI-05) Pittsburgh, Pennsylvania.

[31] Sedighpour, M., Ahmadi, V., Yousefikhoshbakht, M., Didehvar, F. and Rahmati. F. (2013). Solving the open vehicle routing problem by a hybrid ant colony optimization, Kuwait journal of science, 40(3), 20-36.

[32] Sedighpour, M., Yousefikhoshbakht, M. and MahmoodiDarani, N. (2011). An Effective Genetic Algorithm for Solving the Multiple Traveling Salesman Problem, Journal of Optimization in Industrial Engineering, 4(8), 73-79.

[33] Singh, A. and Baghel, A. S. (2009). A new grouping genetic algorithm approach to the multiple traveling salesperson problem, Soft Computing, 13, 95–101.

[34] Sofge, D., Schultz, A. and De Jong, k. (2002). Evolutionary Computational Approaches to Solving the Multiple Traveling Salesman Problem Using a Neighborhood Attractor Schema, Applications of Evolutionary ComputingLecture Notes in Computer Science, 2279, 153-162.

[35] Tang, L., Liu, J., Rong, A. and Yang, Z. (2000). A multiple traveling salesman problem model for hot rolling scheduling in ShangaiBaoshan Iron & Steel Complex. European Journal of Operational Research, 124, 267–282.

[36] Wang, X., Regan, A. C. (2002). Local truckload pickup and delivery with hard time window constraints, Transportation Research Part B, 36, 97-112.

[37] Yadlapalli, S., Malik, W. A., Darbha, S. and Pachter, M. (2009). A Lagrangian-based algorithm for a Multiple Depot, Multiple Traveling Salesmen Problem. Nonlinear Analysis: Real World Applications, 10(4), 1990-1999.

[38] YousefiKhoshbakht, M., Didehvar, F. and Rahmati, F. (2013). Modification of the Ant Colony Optimization for Solving the Multiple Traveling Salesman Problem, Romanian academy section for information science and technology, 16(1), 65-80.

[39] Yousefikhoshbakht, M. and Sedighpour, M. (2012). A Combination of Sweep Algorithm and Elite Ant SYSTEM for Solving the Multiple Traveling Salesman Problem, Proceedings of the Romanian Academy, Series A, 13 (4), 293-303.

[40] YousefiKhoshbakht, M. and Sedighpour, M. (2013). A New Imperialist Competitive Algorithm to Solve the Traveling Salesman Problem, International Journal of Computer Mathematics, 90(7), 1495-1505

[41] Yousefikhoshbakht, M. and Hashemi, N. (2013). Solving the Multiple Traveling Salesman Problem by the ICA, Journal of Industrial Engineering & Management, Accepted paper.

[42] Zhou, W. and Li, Y. (2010). improved genetic algorithm for multiple traveling salesman problem Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference, 1, 493 – 495.