An Efficient Combined Meta-Heuristic Algorithm for Solving the Traveling Salesman Problem

Majid Yousefikhoshbakht
Department of Mathematics, Faculty of Science, Bu-Ali Sina University, Hamedan, Iran,
Tel. +98 81 38380595
khoshbakht@basu.ac.ir

Azam Dolatnejad
Young Researchers & Elite Club, Tehran North Branch,
Islamic Azad University, Tehran, Iran

Abstract
The traveling salesman problem (TSP) is one of the most important NP-hard Problems and probably the most famous and extensively studied problem in the field of combinatorial optimization. In this problem, a salesman is required to visit each of \( n \) given nodes once and only once, starting from any node and returning to the original place of departure. This paper presents an efficient evolutionary optimization algorithm developed through combining imperialist competitive algorithm and lin-kernighan algorithm called (MICALK) in order to solve the TSP. The MICALK is tested on 44 TSP instances involving from 24 to 1655 nodes from the literature so that 26 best known solutions of the benchmark problem are also found by our algorithm. Furthermore, the performance of MICALK is compared with several metaheuristic algorithms, including GA, BA, IBA, ICA, GSAP, ABO, PSO and BCO on 32 instances from TSPLIB. The results indicate that the MICALK performs well and is quite competitive with the above algorithms.

Keywords: Imperialist Competitive Algorithm; NP-hard Problems; Lin-Kernighan Algorithm; Traveling Salesman Problem.

1. Introduction
There are a lot of research studies in the field of logistics, ranging from the assignment problems to complex dynamic routing problems. Among the prominent problems in the field of distribution and logistics, the traveling salesman problem (TSP) has arguably been one of the most widely investigated problems of combinatorial optimization in recent years and there have been many research studies to provide a better solution for this problem (Lawler et al., 1985). A large number of problems like the TSP with pickup and delivery, time windows TSP, multi-depot TSP, multiple TSP, vehicle routing problem and so on have arisen from this problem (YousefiKhoshbakht, Didehvar, & Rahmati, 2014). Besides, it has many applications in dealing with other problems, including the Print press scheduling (Rathinam & Sengupta, 2006), School bus routing problem (Svestka & Huckfeldt, 1973), Interview scheduling (Angel et al., 1972), Hot rolling scheduling (Brummit & Stentz, 1998), Design of global navigation satellite system surveying networks (Tang et al., 2000), etc.

The TSP can be represented by a complete graph \( G = (N, A) \) where \( N \) is the set of nodes, and \( A \) is the set of arcs fully connecting the nodes. Let \( c_{ij} \) be the length of the arc \((i,j)\), which is the distance between nodes \( i \) and \( j \). The TSP is the problem of finding a minimal length Hamiltonian circuit, where a Hamiltonian circuit of a graph \( G \) is a closed tour visiting once and only once all \( n = |N| \) nodes of \( G \), and its length is the sum of the lengths of all the arcs of which it is composed.

There have been important advances in developing exact and heuristic algorithms for solving the TSP. There are several exact algorithms of the TSP such as Cutting Planes algorithm (Laporte & Nobert, 1980), branch-and-cut method (Cordeau, Dell’Amico, & Iori, 2010) and lagrangean relaxation and branch-and-bound algorithm (Mak & Boland, 2007). Since the TSP belongs to the class of NP-complete problems, its solution grows exponentially with the increase in
distribution points. Thus, exact algorithms are not capable of solving problems with large dimensions. On the other hand, heuristics are thought to be more efficient for a complex TSP and have become very popular with researchers. A lot of algorithms have been performed on the TSP including heuristic approaches such as the k-opt approach (Potvin, Lapalme, & Rousseau, 1989), minimum spanning tree (Malik, Rathinam, & Darbha, 2007), self-organizing NN approach (Vakhutinsky & Golden, 1994) and the partitioning approach (Karp, 1977).

Although heuristic methods solve NP-complete problems, they become trapped in local optima and cannot obtain optimum solution. As a result a new kind of algorithm called metaheuristics has been proposed in recent years (Ahmadvand, Yousefi-Khoshbakht, & Mahmoodi-Darani, 2012). These algorithms try to seek high quality solutions while attempting to reduce the computational time. Furthermore, in comparison with heuristic algorithms, because metaheuristics use the randomization concept in search for a solution, this group is more effective in escaping from local optimum and can produce solutions of higher quality. Some of the well-known metaheuristics are the particle swarm optimization (Anantathanavit & Munlin, 2015), neural network (Tarkov, 2015), ant colony optimization (Sedighpour et al., 2013), simulated annealing (Jeong, Kim, & Lee, 2009), neural networks (Jolai & Ghanbari, 2010) and Genetic Reactive Bone Route Algorithm whit Ant Colony System (Yousefikhoshbakht, Malekzadeh, & Sedighpour, 2016).

The ICA is a global search strategy which uses the socio-political competition among empires as a source of inspiration. Like many evolutionary algorithms, the primary ICA may fall into local minimum trap during the search process and it might get far from the global optimum. For this reason, we try to propose an effective two-phase ICA called MICALK by adopting a behavior between rigid regularity and randomness based on pure chance. At the first stage, the TSP is solved by a modified ICA, and at the second stage, Since the Lin-Kernighan algorithm is one of the most effective local search algorithms for solving the TSP, this algorithm is used for improving solutions. In more details, to enhance the global exploration capability, the 2-opt local search with a high probability of $\alpha$ and 0-1 and 1-1 exchanges with low probability $\beta$ and $\mu$ respectively are used. This probability of movement has changed during the search process. The main contributions of the paper are as follows:

1. ICA to enhance the ability of escaping from a local optimum
2. Presenting an effective ICA algorithm for discrete problems
3. Combining the ICA algorithm with a Lin-Kernighan algorithm called MICALK
4. Presenting a new algorithm which is equipped with diversification and intensification mechanisms for solving the TSP.

The rest of this paper is organized as follows: in Section 2 we present the basic principle of ICA, Lin-Kernighan (LK) algorithm and then explain the details of the process of the proposed algorithm. In Section 3, the proposed algorithm will be compared with some of the other algorithms on standard problems which are included in the TSP library. The conclusions and the future works are presented in section 4.

2. Solution Method

In this section at first, the classic ICA and LK are explained and then the hybrid of ICA and LK as the proposed algorithm is explained in more detail. Finally, because the MICALK is metaheuristic, parameter setting is presented.

2.1. The basic principle of ICA

During the last three decades, evolutionary optimization methods inspired by natural processes have shown good performance in solving combinatorial optimization problems. These algorithms have become increasingly popular through the development and utilization of intelligent paradigms in the design of advanced information systems. When the task is optimized within complex domains of data or information, nature-inspired approaches such as swarm or flocking
intelligence (bird flocks or fish schools inspired particle swarm optimization), artificial immune systems which mimic the biological ones, ant colonies (ants’ foraging behaviors gave rise to ant colony optimization), or optimized performance of bees and so on are widely used to solve engineering optimization problems.

ICA, which proved to be capable of obtaining acceptable performance of some of benchmark cost functions is one of the newest algorithms created in the recent decades. This new optimization algorithm was first used by Atashpaz Gargari et al to solve the combinational optimization problems in 2007 (Atashpaz Gargari & Lucas, 2007). In recent years, the ICA has been successfully applied to several NP-hard combinational optimization problems, namely, TSP (YousefiKhoshbakht & Sedighpour, 2013), recommender systems (Sepehri Rad & Lucas, 2008), designing skeletal structures (Kaveh & Talatahari, 2010), and many other optimization problems such as the characterization of elasto-plastic properties of materials, designing optimal layout for factories, adaptive antenna arrays, intelligent recommender systems, and optimal controller for industrial and chemical processes (Shokrollahpour, Zandieh, & Dorri, 2010). This evolutionary optimization strategy has shown great performance in both convergence rate and better global optimum achievement. Furthermore, the proposed evolutionary optimization algorithms are generally inspired by modeling the natural processes and other aspects of species evolution, especially human evolution, which have not been considered up to now. The method proposed in this work uses socio-political evolution of human as a source of inspiration for developing a powerful optimization strategy. What is more important, the ICA considers the imperialism as a level of human social evolution and by mathematically modeling this complicated political and historical process harnesses it as a tool for evolutionary optimization.

The ICA is a novel global search strategy which uses imperialism and imperialistic competition process as a source of inspiration. This algorithm is based on the fact that in a real world, countries try to extend their power over other countries in order to use their resources and bolster their own government. The first step in ICA 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 altogether form some empires. As it is shown in Figure 1 (Atashpaz Gargari & Lucas, 2007), bigger and stronger empires have more colonies than smaller and weaker ones.

![Figure 1. The Initial Empires](image-url)
After initial empires are formed, 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 one of the colonies possesses more power than its relevant imperialist after this movement, 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, weak empires collapse and powerful ones take the possession of their colonies - Figure 2 (Atashpaz Gargari & Lucas, 2007). The empire, which has lost all its colonies, will collapse. At last, the most powerful empire will take the possession of other empires and will win 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. Figure 3 shows the flowchart of the basic ICA.¹

![Figure 2. Eliminate the weakest colony of the weakest empire](https://en.wikipedia.org/wiki/File:Imperialist-competitive-algorithm-flowchart.jpg)

![Figure 3. Flowchart of the ICA](https://en.wikipedia.org/wiki/File:Imperialist-competitive-algorithm-flowchart.jpg)

2.2. Lin-Kernighan Algorithm

The enormous literature on meta-heuristics indicates that a promising approach for obtaining high-quality solutions is to couple a local search algorithm with a mechanism to generate initial solutions. A simple example of this type of algorithm is the 2-opt algorithm. This algorithm starts with a feasible tour and continues by omitting two arcs of the tour, which are not adjacent and then connects them again by another method in such a way that the new tour length is shorter. It can be noted that there are several routes for connecting nodes and producing the tour again, but a state that satisfies the problem’s constraints is acceptable. Omitting and reconnecting two arcs are repeated until no improving 2-opt is found. Besides, the 2-opt algorithm is a unique case of the $\omega$-opt algorithm, where in each step $\omega$ links of the current tour are replaced by $\omega$ links in such a way that a shorter tour is achieved. The number of operations to test all $\omega$-exchanges increases quickly as the number of nodes increases. The time complexity of testing $\omega$-exchange is equal to $O(n^{\omega})$ in which there is no nontrivial upper bound of the number of $\omega$-exchanges.

However, since there are weaknesses, the value of $\omega$ and what $\omega$ to use to achieve the best compromise between running time and quality of the solution must be specified in advance. The Lin-Kernighan algorithm is a generalization of this simple principle form and is one the most effective algorithms for solving the TSP. This algorithm removed this disadvantage by introducing a powerful variable $\omega$-opt algorithm in which the value of $\omega$ is changed during its execution. At each iteration the algorithm considers a growing set of potential exchanges which start with $r = 2$. 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 $\omega$ and the algorithm decides what the value of $\omega$ should be. If an interchange of $\omega$ 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. Our approach

In order to apply the proposed algorithm on the TSP, at the first stage, feasible solutions or primitive countries should be introduced in a way which is compatible with the construction of the mentioned problem. Therefore, a bisection array shown in Figure 4 is used. In this array, the visited nodes are ordered from left to right in the first section and the value of the objective function is shown in the second section.

![Figure 4. A Country in ICA](image)

Therefore, a defined number of primitive solutions ($r$) must be randomly generated and the values of the objective function $f_i$ for each $i=1,...,r$ must be obtained. Then, these solutions and their values are imported to matrix D in which every row shows a primitive solution and its value of the objective function. It should be noted that using a random construction at this level leads to obtaining solutions which have an irregular construction in feasible space. Then, $m$ countries which have better objective function are selected and are called empire countries. Furthermore, the number of colonies devoted to each empire $j$ is calculated by the formula (1).

$$k_j = \text{int}\left((1 - \frac{f_j}{\sum_{i=1}^{m} f_i}) \times (r - m)\right)$$  (1)

This formula leads that more colonies are allocated to empires with fewer objective functions and as a result a bigger empire is formed. The Int function used in the formula is the floor
function which causes the allocation of an integer number of colonies to each empire. It should be noted that allocating colonies to empires is conducted randomly and if some countries might not belong to any empire due to the property of the Int function, these countries are allocated to the most powerful empire. After the empires are formed, 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, we must use an absorption function which includes a randomization concept so that the results of combinations will not yield a very similar response. To achieve this goal, we have utilized a novel and innovative method. As an example, first, two possible responses [5 2 9 4 8 3 1 6 7] and [9 2 7 6 5 3 1 8 4] are considered as imperialist and colony respectively. Then, a random number between 2 and n-1 is selected (n is the number nodes of instance and is 9 here). After that, some nodes equal to the selected number are chosen from the colonies and are arranged according to the order of the imperialist countries. If the selected random number between 1 and 8 is 4, then 4 nodes are randomly chosen from the colonies like 2, 6, 5, and 3. Then, their order in imperialist country which is 5, 2, 3, and 6 in the example is found. Therefore, the new result for the colony will be [9 5 7 2 3 6 1 8 4]. The absorption function is performed for all colonies in comparison to imperialist countries and the results and the values are replaced with the best results and the values obtained in the current iteration provided that the new results are better.

At the next stage, $p$ percent of countries experience a revolution. This causes variations in colonies in each empire and if possible their quality increases at each stage. The proposed methods for this stage are the 0–1 Exchange move. In this move, a candidate node is removed from its original route and is inserted in the best position. The replacement is done if the new results are better than the previous ones.

Most successful meta-heuristic methods have paid attention to global search and search in the whole solution space as far as possible. As the algorithm proceeds, it moves to better solutions and the global search switches to a local search. We have factored in this issue too, and have represented the probability of 0-1, 1-1- and 2-opt move with $\alpha, \beta$ and $\mu$ respectively so that $\alpha + \beta + \mu = 1$. In the 1-1-Exchange, two nodes are randomly selected and exchanged with each other. Finally, in the 2-Opt move, two non-adjacent edges are replaced by two other edges. It should be noted that there are several routes for connecting nodes and producing the tour again, but a state which satisfies the problem’s constraints is acceptable. Note that although 2-opt local search as a powerful global search algorithm is more used at the beginning of algorithm for global search, 0-1 and 1-1 exchanges are more applied at the end of the algorithm because these algorithms might lead to premature convergence to suboptimal regions. In other words, before the algorithm finishes a complete global search, it tends to adopt a local search and consequently relatively weak results are attained. Therefore, whenever the algorithm continues the probability of $\alpha$ decreases and the probability of $\beta$ and $\mu$ increases. Adding this behavior to the imperialist algorithm revolution policy leads to creating the proper conditions for the algorithm to escape from local peaks. Thus, as mentioned before, the probability of using the 2-opt, 0-1 and 1-1 exchanges at the first step of the proposed algorithm is considered 0.50, 0.25 and 0.25 and then during the other steps of the proposed algorithm, they are gradually converted to 0.20, 0.40 and 0.40.

After the results and the values are calculated for all colonies, these countries might have a better objective value compared to their respective imperialists. Therefore, a colony with the best value in each empire is chosen and if it possesses a better objective value, it replaces an imperialist country. If there are a few colonies with the same objective value, 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 change 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 are replaced, the best results and values 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, part 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 2.

$$h_j = f_j + \lambda s_j \quad j = 1, \ldots, m$$  \hspace{1cm} (2)

In this formula $h_j, s_j, \text{ and } \lambda$ 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. At this stage, the final condition is checked and if it is met, the algorithm ends. Otherwise, the algorithm is iterated by returning to the absorption function step. To end the proposed algorithm, one of the two conditions must be met: the iteration of algorithm $n$ times or the survival of just one empire. These conditions are checked at the end of each algorithm iteration. If any one 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.

Moreover, in order to prevent the ICA from getting trapped in stagnation, we used a local searching algorithm 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 have of course their pros and cons. Since Lin-Kernighan algorithm is simple and it is one of the most successful methods for generating optimal or near optimal solutions for the TSP, we have used it in this study. The main steps of MICALK are summarized in the pseudo-code given in Figure 5.

---

**Figure 5. The process of MICALK for solving the TSP**

**Step 1:** Generate $r$ random solutions of the TSP and initialize the empires.

**Step 2:** Move the colonies toward their relevant imperialist by the proposed absorption function.

**Step 3:** Change $p$ percent of colonies as revolution by the 0-1 Exchange move.

**Step 4:** If there is a colony in an empire which has better cost than the imperialist, exchange the positions of that colony and the imperialist.

**Step 5:** Compute the total cost of all empires by formula (2).

**Step 6:** Pick the weakest colony from the weakest empires and give it to the best empire (Imperialistic competition).

**Step 7:** Eliminate the powerless empires.

**Step 8:** If the quality of the best solution is increased in this iteration, apply lin-kernighan algorithm to $s$ and save the best so far solution.

**Step 9:** If the iteration of algorithm reaches to $n$ times or the just one empire is reminded, stop, if not go to 2.
2.4 Parameter settings

The proposed MICALK algorithm contains several parameters like other metaheuristics for solving the optimization problems. The quality solutions produced by the MICALK have been dependent on the different values of the user-controlled parameters of the algorithm. In this algorithm, three important parameters exist and their values directly or indirectly affect the performance of the algorithm. For each of the benchmark instances, ten different runs with the selected parameters were performed after the selection of the final parameters. In general, it is not easy to obtain the best combination of algorithm parameters, but a parameter setting procedure is necessary to reach the best balance between the quality of the obtained solutions. These parameters are selected in this paper after thorough testing. All of the parameter values have been determined on the Eil101by the numerical experiments so that several alternative values for each parameter were tested while all the other values were held constant. It should be noted that only parameters which gave the best computational results concerning the quality of the solution were selected. The details regarding the settings of the parameters and their values are explained in Table 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Candidate Value</th>
<th>Best Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>Number of primitive countries</td>
<td>100, 200, 300, 400, 500</td>
<td>400</td>
</tr>
<tr>
<td>$m$</td>
<td>Number of Imperialist countries</td>
<td>Int $[r/2, r/3, r/4, r/5, r/6]$</td>
<td>Int $[r/4]$</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>The impact coefficient, which determines the relative power of a colony compared to an empire</td>
<td>0.1, 0.2, 0.3, 0.4, 0.5</td>
<td>0.2</td>
</tr>
</tbody>
</table>

3. Computational experiments

In this section, first, the algorithms tested on 33 classic benchmark problems for the TSP are presented and then some numerical results of the comparison between the proposed algorithm and some metaheuristic algorithms are given. These algorithms are applied and tested on several Euclidean sample instances of TSPLIB with sizes ranging from 24 to 318 nodes. Because the proposed approach is a metaheuristic algorithm, the results are reported for ten independent runs in which the algorithm was executed until the best solution was iterated 10 times.

The algorithms are coded by Matlab language and implemented on a Pentium 4 PC at 3GHZ (1GB RAM). The parameters of the proposed algorithm are selected after thorough testing. A number of different alternative values were tested and the ones selected are those which yielded the best computational results concerning both the quality of the solution and the computational time needed to achieve this solution. Thus, the selected parameters are:

- Number of countries equal to $p$
- Number of empires equal to $N/5$
- $\lambda$ is equal to 0.3.

Table 2 shows the results of the proposed algorithm for the TSP benchmark problem instances. In this table, columns 2-6 show the problem size $n$, the best value result of MICALK (BVR), the worst value result of MICALK (WVR), the average value of MICALK (AV) over the ten runs for each problem, and the best known solutions (BKS), respectively. The seventh column contains the CPU time for each problem ($Opt$), and the eighth for the relative deviation ($RD$) where the relative deviation is calculated as $(MICALK - Opt)/Opt \times 100\%$, where MICALK denotes the cost of the optimum found by the proposed algorithm, and $Opt$ is the cost of the optimal solution.
This table shows that the MICALK can be used to solve the TSP effectively. As table 2 indicates, the maximum relative error and average relative error for 33 test problems are 1.57% and 0.21%, respectively.

Figure 6 shows some of the best solutions searched by the proposed method. In this figure, the horizontal axis represents the x-axis with increasing positive values to the right and the vertical axis represents the y-axis with increasing positive values upward.
Table 3 shows the results obtained for the second problem instances and presents the comparison of the best results of our algorithm with other published research studies, including genetic algorithm (GA) (Ray, Bandyopadhyay, & Pal, 2004), particle swarm optimization (PSO) (Zhong, Zhang, & Chen, 2007), bee colony optimization (BCO) (Wong, Low, & Chong, 2008), african buffalo optimization (ABO) (Odili & Mohmad Kahar, 2016), genetic simulated annealing ant colony system with particle swarm optimization (GSAP) (Chen & Chien, 2011), ICA (YousefiKhoshbakht & Sedighpour, 2013), bat algorithm (BA) (Osaba et al., 2016) and improved bat algorithm (IBA) (Osaba et al., 2016) in terms of the optimal solution found. The results of this comparison show that the proposed algorithm gains equal solutions with the GA in GR24, Bayg29 and GR48 which are not large scale problems and it gains better solutions than the GA in scale problems such as St70 and KroA100. Furthermore, the results indicate that although the ICA provides a better solution and equal solutions compared with MICALK for seventeen instances, this algorithm obtains worse solutions than MICA for fourteen solutions.

Figure 6. Some best routes found by the proposed algorithm
Table 3. Comparison between MICALK and other metaheuristic algorithms

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GR24</td>
<td>24</td>
<td>-</td>
<td>-</td>
<td>1272</td>
<td>1272</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1272</td>
<td>1272</td>
</tr>
<tr>
<td>Bayg29</td>
<td>29</td>
<td>-</td>
<td>-</td>
<td>1610</td>
<td>1610</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1610</td>
<td>1610</td>
</tr>
<tr>
<td>GR48</td>
<td>48</td>
<td>-</td>
<td>-</td>
<td>5046</td>
<td>5046</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5046</td>
<td>5046</td>
</tr>
<tr>
<td>ATT48</td>
<td>48</td>
<td>-</td>
<td>-</td>
<td>10628</td>
<td>10628</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10661</td>
<td>10628</td>
</tr>
<tr>
<td>Eil51</td>
<td>51</td>
<td>426</td>
<td>430</td>
<td>426</td>
<td>427</td>
<td>426</td>
<td>427</td>
<td>428</td>
<td>426</td>
<td>426</td>
<td>426</td>
</tr>
<tr>
<td>Berlin52</td>
<td>52</td>
<td>7542</td>
<td>7676</td>
<td>7542</td>
<td>7542</td>
<td>-</td>
<td>7542</td>
<td>-</td>
<td>7542</td>
<td>7542</td>
<td>7542</td>
</tr>
<tr>
<td>ST70</td>
<td>70</td>
<td>675</td>
<td>696</td>
<td>685</td>
<td>677</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>675</td>
<td>675</td>
</tr>
<tr>
<td>Eil76</td>
<td>76</td>
<td>539</td>
<td>559</td>
<td>-</td>
<td>538</td>
<td>538</td>
<td>538</td>
<td>540</td>
<td>539</td>
<td>538</td>
<td>538</td>
</tr>
<tr>
<td>KroA100</td>
<td>100</td>
<td>21282</td>
<td>21884</td>
<td>21504</td>
<td>21282</td>
<td>21282</td>
<td>21311</td>
<td>21296</td>
<td>21763</td>
<td>21282</td>
<td>21282</td>
</tr>
<tr>
<td>KroB100</td>
<td>100</td>
<td>22141</td>
<td>22843</td>
<td>-</td>
<td>22141</td>
<td>22141</td>
<td>22160</td>
<td>-</td>
<td>22637</td>
<td>22141</td>
<td>22141</td>
</tr>
<tr>
<td>KroC100</td>
<td>100</td>
<td>20749</td>
<td>20802</td>
<td>-</td>
<td>20749</td>
<td>20749</td>
<td>20755</td>
<td>-</td>
<td>20853</td>
<td>20749</td>
<td>20749</td>
</tr>
<tr>
<td>KroD100</td>
<td>100</td>
<td>21294</td>
<td>21727</td>
<td>-</td>
<td>21294</td>
<td>21309</td>
<td>21347</td>
<td>-</td>
<td>21643</td>
<td>21389</td>
<td>21294</td>
</tr>
<tr>
<td>KroE100</td>
<td>100</td>
<td>22068</td>
<td>22323</td>
<td>-</td>
<td>22068</td>
<td>22068</td>
<td>22088</td>
<td>-</td>
<td>22450</td>
<td>22068</td>
<td>22068</td>
</tr>
<tr>
<td>Eil101</td>
<td>101</td>
<td>646</td>
<td>667</td>
<td>-</td>
<td>629</td>
<td>630</td>
<td>-</td>
<td>-</td>
<td>635</td>
<td>629</td>
<td>629</td>
</tr>
<tr>
<td>Lin105</td>
<td>105</td>
<td>-</td>
<td>-</td>
<td>14379</td>
<td>14379</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>15288</td>
<td>14379</td>
<td>14379</td>
</tr>
<tr>
<td>KroA150</td>
<td>150</td>
<td>-</td>
<td>-</td>
<td>26524</td>
<td>26524</td>
<td>26526</td>
<td>-</td>
<td>27858</td>
<td>26524</td>
<td>26524</td>
<td></td>
</tr>
<tr>
<td>KroB150</td>
<td>150</td>
<td>-</td>
<td>-</td>
<td>26154</td>
<td>26130</td>
<td>26169</td>
<td>-</td>
<td>26535</td>
<td>26130</td>
<td>26130</td>
<td></td>
</tr>
<tr>
<td>KroA200</td>
<td>200</td>
<td>-</td>
<td>-</td>
<td>29379</td>
<td>29383</td>
<td>29370</td>
<td>29563</td>
<td>29961</td>
<td>29368</td>
<td>29368</td>
<td></td>
</tr>
<tr>
<td>KroB200</td>
<td>200</td>
<td>-</td>
<td>-</td>
<td>29437</td>
<td>29451</td>
<td>29487</td>
<td>-</td>
<td>30350</td>
<td>29487</td>
<td>29437</td>
<td></td>
</tr>
<tr>
<td>Pr107</td>
<td>107</td>
<td>44794</td>
<td>44618</td>
<td>-</td>
<td>44303</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>44303</td>
<td>44303</td>
<td>44303</td>
</tr>
<tr>
<td>Pr124</td>
<td>124</td>
<td>59412</td>
<td>59627</td>
<td>-</td>
<td>59087</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>59076</td>
<td>59030</td>
<td>59030</td>
</tr>
<tr>
<td>Pr136</td>
<td>136</td>
<td>99351</td>
<td>101631</td>
<td>-</td>
<td>96845</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>93812</td>
<td>96772</td>
<td>96772</td>
</tr>
<tr>
<td>Pr144</td>
<td>144</td>
<td>58537</td>
<td>58588</td>
<td>-</td>
<td>58537</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>58537</td>
<td>58537</td>
<td>58537</td>
</tr>
<tr>
<td>Pr152</td>
<td>152</td>
<td>73921</td>
<td>74172</td>
<td>-</td>
<td>73682</td>
<td>73730</td>
<td>-</td>
<td>73682</td>
<td>73682</td>
<td>73682</td>
<td></td>
</tr>
<tr>
<td>Pr264</td>
<td>264</td>
<td>49756</td>
<td>50256</td>
<td>-</td>
<td>50111</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>49765</td>
<td>49135</td>
<td>49135</td>
</tr>
<tr>
<td>Pr299</td>
<td>299</td>
<td>48310</td>
<td>49142</td>
<td>-</td>
<td>49879</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>48299</td>
<td>48191</td>
<td>48191</td>
</tr>
<tr>
<td>Li318</td>
<td>318</td>
<td>-</td>
<td>-</td>
<td>42487</td>
<td>42487</td>
<td>-</td>
<td>-</td>
<td>44685</td>
<td>42383</td>
<td>42029</td>
<td></td>
</tr>
<tr>
<td>Rat755</td>
<td>575</td>
<td>-</td>
<td>-</td>
<td>6823</td>
<td>6891</td>
<td>6774</td>
<td>-</td>
<td>-</td>
<td>6773</td>
<td>6773</td>
<td>6773</td>
</tr>
<tr>
<td>Rat783</td>
<td>783</td>
<td>-</td>
<td>-</td>
<td>8905</td>
<td>8988</td>
<td>8811</td>
<td>-</td>
<td>-</td>
<td>8811</td>
<td>8806</td>
<td>8806</td>
</tr>
<tr>
<td>Rl1323</td>
<td>1323</td>
<td>-</td>
<td>-</td>
<td>272341</td>
<td>277642</td>
<td>270480</td>
<td>-</td>
<td>-</td>
<td>270456</td>
<td>270199</td>
<td></td>
</tr>
<tr>
<td>Fi1400</td>
<td>1400</td>
<td>-</td>
<td>-</td>
<td>20453</td>
<td>20593</td>
<td>20134</td>
<td>-</td>
<td>-</td>
<td>20132</td>
<td>20127</td>
<td></td>
</tr>
<tr>
<td>D1655</td>
<td>1655</td>
<td>-</td>
<td>-</td>
<td>64521</td>
<td>64151</td>
<td>62128</td>
<td>-</td>
<td>-</td>
<td>62128</td>
<td>62128</td>
<td></td>
</tr>
</tbody>
</table>
From the comparison between IBA and the proposed algorithm, it can be seen that IBA in nine examples has been able to find the optimum and in eight examples can yield to find solutions with gap of less than 1 percent. However, the proposed MICALK has found better solutions than IBA for larger size of instances. The last powerful algorithm compared to the proposed is IBA. This algorithm can obtain nine out of twenty best solutions, but the MICALK can yield better solutions than IBA in the remaining 11 examples. It should be noted that three reminded algorithms have had a weak performance in general and have not been able to produce the best solutions in most of the examples. In more details, the computational experiments confirm that in general the proposed algorithm provides better results compared to PSO, BA and BCO algorithms in terms of the quality of the solutions and is able to find the best solutions for twenty three out of thirty two in this table.

The evolution of the best solution found by the proposed algorithm is plotted in Figure 7 during a typical execution when solving instance Eil51 and KroB100. In this figure, the horizontal and vertical axes show the number of iterations and gained values of the proposed algorithm respectively. Besides, there is a fast convergence toward the BKS at the beginning of the execution while in the rest of the search the evolution of the BKS is not that fast.

4. Conclusion

In this paper, a hybrid algorithm called MICALK which combines ICA and Lin-Kernighan algorithm was proposed for solving the TSP. We have also done experiments using two different data sets of TSP instances, including 44 problems with 24–1655 nodes from the TSPLIB. The experimental results indicate that the gap of the proposed algorithm stays on average below 1.6% of the execution time and the MICALK uniformly produces higher performance solutions compared with other competing metaheuristics including GA, BA, IBA, GSAP, ABO, PSO, ICA and BCO on the TSP. It seems that the combination of the proposed algorithm with ant colony system or tabu search will yield better results for large problems of TSP. Applying this method in other combinational optimization problems, including the multiple traveling salesmen problem, vehicle routing problem, School bus routing problem and the sequencing of jobs is suggested for future research studies.

References


Rathinam, S. & Sengupta, R. (2006). Lower and upper bounds for a symmetric Multiple Depot, 
*Multiple Travelling Salesman Problem*, Research Report, UCB-ITSRR- 2006-2, Institute of 
Transportation Studies.


hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex,” *European Journal of 


traveling salesman problem with FIFO loading,” *Computers & Operations Research* 37(5): 
970-980.

Mak, V. & Boland, N. (2007). “Polyhedral results and exact algorithms for the asymmetric 
travelling salesman problem with replenishment arcs,” *Discrete Applied Mathematics* 155(16): 
2093-2110.


Generalized Multiple Depot,” *Multiple Travelling Salesman Problem. Operations Research 
Letters* 35(6): 747-753,

*Proceedings of the IEEE international conference on neural network*, 4535–4540,


Salesman Problem by an Efficient Hybrid Metaheuristic Algorithm, Journal of Advances in 
Computer Research, 3 (3), 75-84.


Tarkov, M. S. (2015). Solving the traveling salesman problem using a recurrent neural 

Sedighpour, M., Ahmadi, V., YousefiKhoshbakht, M., Didehvar, F., & Rahmati, F. (2013). Solving the 
Open Vehicle Routing Problem by a Hybrid Ant Colony Optimization, *Kuwait Journal of 
Science*, 41 (3), 139-162


networks for solving travelling salesman problem,” *Expert Systems with Applications* 37(7), 
5331-5335.

Salesman Problem Based on The Genetic Reactive Bone Route Algorithm whith Ant Colony 

Optimization Inspired by Imperialistic Competition, *IEEE Congress on Evolutionary 
Computation (CEC 2007)*, 4661 – 4667.


