Solving Euclidean Minimal Spanning Tree Problem Using a New Meta-heuristic Approach: Imperialist Competitive Algorithm (ICA)

S. M. Hosseini[1], A. A. Khaled[2], M. Jin[3]
[1] Department of Industrial and Systems Engineering, Mississippi State University, Mississippi State, USA
[2]Department of Industrial and Systems Engineering, Mississippi State University, Mississippi State, USA
[3]Department of Industrial and Systems Engineering, The University of Tennessee, Knoxville, USA
([1]s.m.hosseini48@gmail.com, [2]ak511@msstate.edu, [3]jin@utk.edu)

Abstract – **The minimum spanning tree (MST) problem has numerous applications in design of communication, computer and transportation networks. This paper proposes an application of a new meta-heuristic called Imperialist Competitive algorithm (ICA) for solving a special case of MST defined on a Euclidean plane, called the Euclidean minimum spanning tree (EMST) problem. ICA is inspired by human's socio-political evolution. The solution quality and speed of the proposed are verified by numerical examples compared to a commercial optimization solver.**

Keywords - **Spanning tree, ICA, EMST**

I. INTRODUCTION

The minimum spanning tree (MST) problem is among the most important problems in the graph theory with numerous applications, such as transportation network design and data transferring in a communication network. The problem is defined on a graph $G = (V, E)$, where V is the vertex set with $|V| = v$ and E is the undirected edge set with $|E| = m$. Every edge $e \in E$ is associated with non-negative cost c_e. A minimum spanning tree is defined as a connected sub graph of G having no cycle and covering all vertices in V with the minimum cost. The MST problem can be formulated as the following integer program (IP).

$$\text{Minimize} \quad \sum_{e \in E} c_e x_e \tag{1}$$

S.T.

$$\sum_{e \in E} x_e = v - 1 \tag{2}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \qquad \forall S \subset V, S \neq \emptyset \tag{3}$$

$$x_e \in \{0, 1\}. \tag{4}$$

The objective function (1) minimizes the sum of costs of the edges in the selected spanning tree. Constraint (2) refers to a cardinality constraint ensuring that we choose exactly $v - 1$ edges. Constraint set (3) is known as the sub-tour elimination constraints implying that the set of chosen edges contain no cycles. $E(S)$ is the set of edges in G whose both ends belong to the vertex set S. The binary decision variable x_e in (4) takes value 1 if edge e is selected and 0 otherwise.

Many researches have been conducted to investigate the MST problem. The first algorithm was introduced by Boruvka in 1926 [1] and has the computational complexity of $O(m \log v)$. Two most well-known algorithms are greedy algorithms developed by Kruskal and by Prim with computational complexity of $O(m \log v)$ and $O(m + v \log v)$ respectively [2]. Neuman and Witt proposed a heuristic based on ant colony optimization (ACO) using the random walk concept [3].

This paper deals with a special class of the MST problem in which all pairs of vertices in a Euclidean plane are connected with an edge with the cost associated to the Euclidean distance. It is known as the Euclidean minimum spanning tree (EMST) problem. The most straightforward algorithm for the EMST problem with v vertices actually constructs the complete graph containing $v(v-1)/2$ edges where each edge cost is computed by finding the distance between each pair of points and then runs either Prim's or Krushal's algorithm [4]. Earlier works on EMST can be found in [5-7]. More recently, March *et al.* [8] introduced a fast and general EMST algorithm based on clustering and analysis of astronomical data. The running time of all above work on EMST is equivalent to $O(v \log v)$.

Traditional approaches for solving the EMST problem scale quadratically and their computational time increases fast when the number of vertices and edges increases. Hence, meta-heuristic approaches could be appropriate alternatives for solving EMST problems with a large number of vertices and edges. This paper proposes an algorithm based on a relatively new meta-heuristic procedure called imperialist competitive algorithm (ICA).

The remainder of the paper is organized as follows. Section II briefly introduces ICA. Section III describes the detailed procedure for implementing ICA to solve the EMST problem with two numerical examples. Finally, Section IV concludes the paper and discusses future research directions.

II. IMPERIALIST COMPETITIVE ALGORITHM

ICA is a new population-based heuristic algorithm inspired by human's socio-political evolution [9]. Although this is a new meta-heuristic approach, its application is widespread in solving many complex optimization problems such as job-shop scheduling, vehicle routing, traveling salesman, and quadratic assignment problems. Some recent works on the application of ICA can be found in [10-12]. Their results indicate that the run time of ICA is averagely less than other algorithms.

ICA starts with an initial population in analogy with most of the evolutionary algorithms (*e.g.*, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), ACO). Each population individual in ICA is called a *country*, equivalent to a particle in PSO and a chromosome in GA. Each country is associated with a cost. In the initialization step for a minimization problem, some of the countries with the least cost in the population are selected as *imperialists*, and then the rest of population, called *colonies*, is divided and allocated to imperialists based on their powers. The power of each country is defined as an inverse of its cost. A collection of an imperialist and its colonies form an *empire*. After forming initial empires, the colonies in each empire start moving toward their relevant imperialist country. This process is known as the *assimilation* process, illustrated in Fig. 1, where the imperialist country absorbs its colony in the language and culture axes. d represents the distance between the colony and its imperialist. The colony in Fig. 1 moves toward the imperialist country to the new position along with extent of x. Here, x is a random variable with a uniform distribution and can be represented as

$$x \sim U(0, \beta \times d). \tag{5}$$

The value of β is typically greater than 1. The human history of the imperialism indicates that the assimilation cannot be performed desirably. Therefore, a random deviation parameter θ is considered in the direction of a colony movement. The variation range of θ is determined by an arbitrary parameter γ. Large values of γ facilitate the global diversification while smaller values encourage local intensification. In Fig. 1, the value of θ is usually expressed in terms of radian and also determined using a uniform distribution given in (6). In most of implementations the values of β and γ at about 2 and $\pi/4$ respectively result in good convergence of ICA to the global minimum.

$$\theta \sim U(-\gamma, \gamma). \tag{6}$$

Similar to the mutation process in GA, ICA introduces a sudden change in socio-political characteristic of a country, known as *revolution*. In the optimization domain, a revolution means that a solution jumps randomly over the space to control the global diversification of countries. Revolutions prevent the algorithm's convergence to local optima at early iterations. Fig. 2 shows a revolution event.

After assimilation and revolution processes, the costs of colonies in an empire are compared with the imperialist in the empire. If there is any colony with lower cost, the colony with the lowest cost and the imperialist exchange their positions. Fig. 3 shows the exchange process between a colony and the imperialist in an empire.

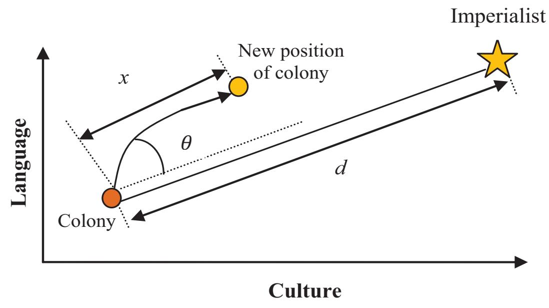


Fig. 1. Movement of colony toward its relevant imperialist [9].

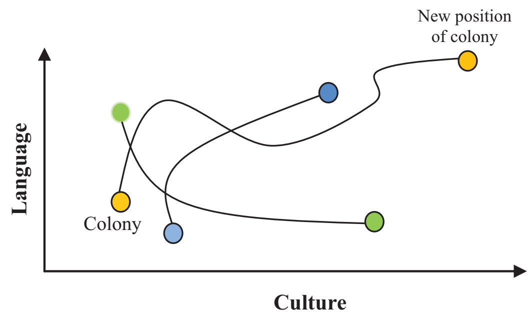


Fig. 2. Revolution process of countries [9].

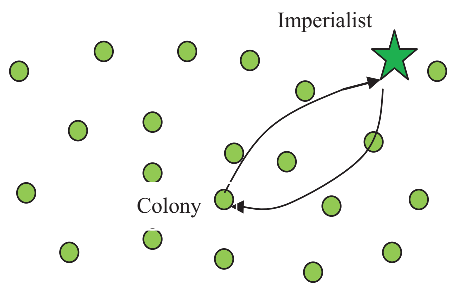


Fig. 3. Exchange positions of imperialist and colony in an empire [9].

The total power of an empire is determined based on the cost of its imperialist plus part of the mean cost of its colonies, as shown in (7).

$$TC_n = cost(imperialist_n) + \delta \times mean\ cost(colonies\ of\ empire_n). \tag{7}$$

TC_n represents the total cost of the n^{th} empire out of the total N empires and δ is a non-negative number smaller than one. When δ approaches zero, the total empire cost will be dominated by the cost of its imperialist. Empirical studies in the literature show that $\delta = 0.1$ often result in good solutions.

The most important part of ICA is the competition among imperialists. During the competition process, all empires try to capture colonies of other empires. This imperialist competition gradually decreases the power of weaker empires and increases that of powerful ones. This fact can be modeled by assigning the weakest colony of the weakest empire to another empire. The likelihood of taking possession of a colony by an empire is determined based on the empire's own total power. Fig. 4 illustrates the imperialist competition process.

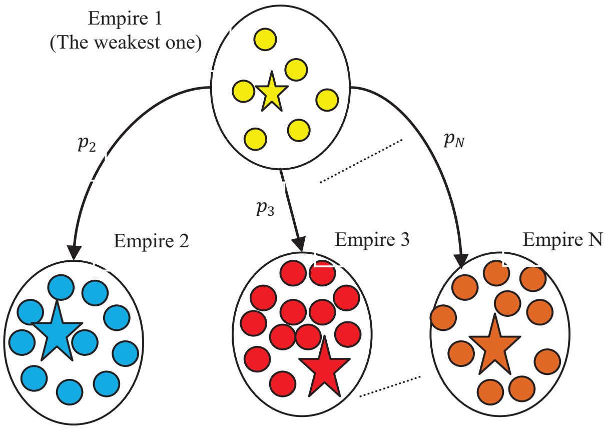


Fig. 4. Empire competition to capture the weakest colony [9].

To obtain the possession probability of each empire, the total cost of each empire n is normalized as in (8).

$$NTC_n = TC_n - \max_{l=1,\dots,N}\{TC_l\}. \tag{8}$$

Here, TC_n and NTC_n represent total and normalized cost of the n^{th} empire respectively. The probability of capturing the weakest colony by the n^{th} empire is computed by (9).

$$p_n = \left|\frac{NTC_n}{\sum_{l=1}^{N} NTC_l}\right|. \tag{9}$$

The capturing probabilities of empires are represented by a $1 \times N$ vector $\boldsymbol{P}$ as in (10).

$$\boldsymbol{P} = [p_1, p_2, \dots, p_N]. \tag{10}$$

Another vector $\boldsymbol{R}$ with the same size as $\boldsymbol{P}$ is created with elements generated by a uniform distribution in (11).

$$\boldsymbol{R} = [r_1, r_2, \dots, r_N]. \tag{11}$$

Vector $\boldsymbol{D}$ is created by subtracting $\boldsymbol{R}$ from $\boldsymbol{P}$ as (12) to add more randomness in the algorithm.

$$\boldsymbol{D} = \boldsymbol{P} - \boldsymbol{R} = [p_1 - r_1, p_2 - r_2, \dots, p_N - r_N]. \tag{12}$$

Based on vector $\boldsymbol{D}$, we assign the weakest colony to the empire that has the largest value in $\boldsymbol{D}$. The advantage of this mechanism over the roulette wheel rule used in GA is that we do not need to calculate any cumulative density functions so that the imperialist competition mechanism has less computational burden than the roulette wheel rule. During the imperialist competition, powerless empires gradually lose their colonies and will be collapsed over time. Finally all colonies will be captured and controlled by the most powerful empire, which is unique. At this point, ICA converges to an ideal solution, which is the only imperialist in the population, and can be considered as a stopping criterion. In addition, the maximum number of iterations can also be used as a stopping criterion. The flowchart of ICA is shown in Fig. 5.

III. IMPLEMENTION OF ICA TO EMST

This section describes the details of implementing ICA to the EMST problem.

A. Representation of the Countries

Each country is represented by a vector of size m, which is the number of decision variables in model (1-4) and the number edges in G. For instance, the graph with $v = 4$ vertices and $m = 5$ edges in Fig. 6 along with its adjacency matrix A, each country is a vector of size 5. All entries in a country vector take values of 0 or 1.

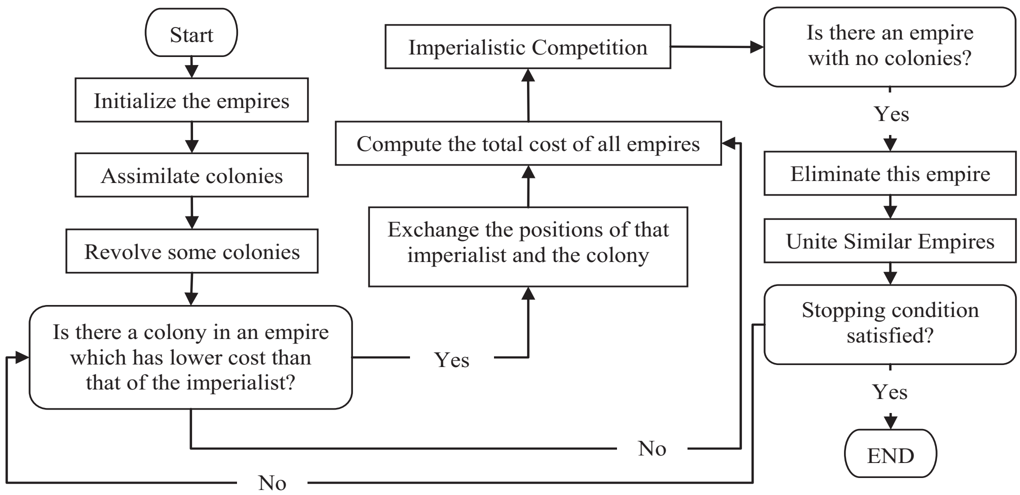
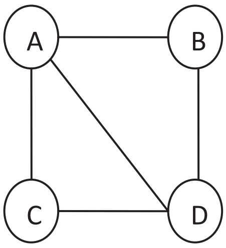


Fig. 5. Flowchart of Imperialist Competitive Algorithm (ICA) [9].



$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Fig. 6. Example graph with four vertices and its adjacency matrix

B. The Cost Function

To apply ICA to the EMST problem, the cost function $\tilde{z}$ for a given solution (country) is defined in (13). The cost function includes a penalty to those solutions that violate the connectivity (feasibility) condition (3) in addition to edge costs. The objective of the EMST problem is to find the solution to minimize $\tilde{z}$.

$$\tilde{z} = \sum_{e \in E} c_e x_e + \alpha q. \tag{13}$$

Here, c_e is the Euclidian distance between the two ends (i, j) of e as (14).

$$c_e = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \tag{14}$$

Here, (x_i, y_i) is the coordinates of vertex i. In (13), α denotes a large enough penalty rate for solution disconnectivity and q is a parameter measuring disconnectivity of the graph formed by edges with $x_e = 1$ in the current solution. In order to calculate q, we review the Connectivity Theorem in the graph theory below.

Connectivity Theorem: Let $G = (V, E)$ be a graph and A be its adjacency matrix. G is strongly connected if and only if matrix B defined in (15) has no zero entries, where

$$B = I + A + \cdots + A^{v-1}. \tag{15}$$

Proof: Let G be a graph with v vertices. a_{ij} and $a_{ij}^{(2)}$ are the ij^{th} entries of $A(G)$ and $A^2(G)$ respectively. Based on the rules of matrix multiplication, we have

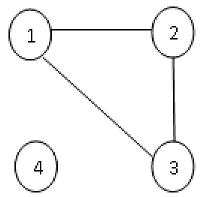
$$a_{ij}^{(2)} = a_{i1}a_{1j} + a_{i2}a_{2j} + \cdots + a_{iv}a_{vj}. \tag{16}$$

$a_{ij}^{(2)}$ is the total possible walks with length 2 from vertex i to vertex j. Similarly, $a_{ij}^{(k)}$ as the ij^{th} element of matrix $A^k(G)$ is the total possible walks from vertex i to vertex j with length k, where $1 \leq k \leq v-1$. Therefore, b_{ij}, the ij^{th} element of B, is the total number of walks with length of less than v from vertex i to vertex j. G is called strongly connected if and only if there are walks with length small than v for each pair (i, j) in G. ■

Based on the Connectivity Theorem, q is defined in (17).

$$q = \frac{\text{number of zeros in } B}{v^2}. \tag{17}$$

Fig. 7 shows an example graph G with 4 vertices and 3 edges along with its adjacency matrix A and matrix B. Matrix B has six zero entries, indicating that graph G is not strongly connected. The disconnectivity percentage of G is $q = \frac{6}{16} = 37.5\%$.



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 5 & 5 & 5 & 0 \\ 5 & 5 & 5 & 0 \\ 5 & 5 & 5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Fig. 7. Example graph for disconnectivity measure q

C. Assimilation process

As mentioned earlier, the assimilation process let the colonies move toward the imperialist through a continuous search while EMST is defined as a discrete problem. To keep all variables binary, we round x_e up to 1 if $x_e \geq 0.5$ in a solution (country) after the assimilation process. Otherwise, x_e is rounded down to 0.5. This may cause a solution (country) violates constraint (2). However, a solution with $\sum_{e \in E} x_e < v - 1$ is penalized in the cost function (13) due to the disconnectivity while a solution with $\sum_{e \in E} x_e > v - 1$ is penalized by the additional cost of including too many edges in the first term of (13). Therefore, any infeasible solution has larger cost and become a weaker country in ICA.

D. Parameters tuning

The performance of ICA depends on how to tune the ICA parameters. We have implemented the parameters tuning experiment for the second example in Subsection *E*. This experimentation is achieved by two steps:

Step 1) *Setting the number of iterations:* We let the number of iterations vary from 200 to 1,200. As shown in Fig. 8, after the 800th iteration, there is no remarkable improvement on cost, so the maximum number of iterations is set to be 800.

Step 2) *Determining the number of countries:* The numbers of countries are tested from 60 to 180 with an increment of 20. Fig. 9 shows that the number of countries can be set at 140 since there is no significant improvement beyond this point.

Other ICA parameters are set as follows: the number of imperialist countries is considered as 10% of total countries (14 imperialist for Example 2). Mean cost coefficient (δ) and assimilation coefficient (β) of colonies are set to 0.1 and 2 respectively.

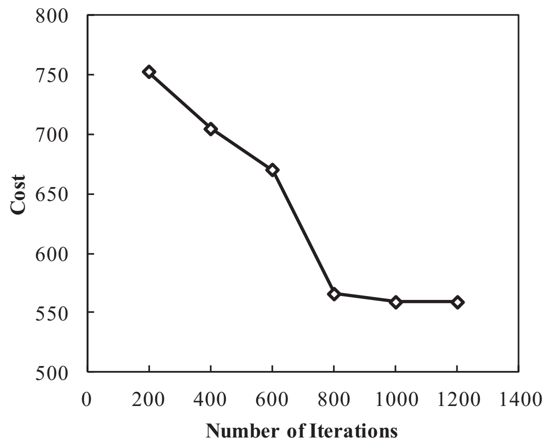


Fig. 8. The impact of ICA's number of iterations on generated solution.

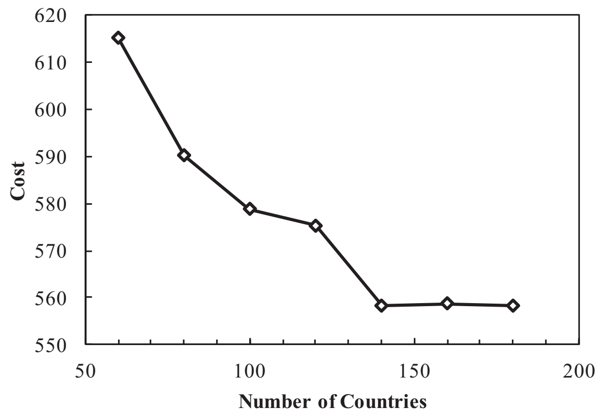


Fig. 9. The impact of ICA's number of countries on generated solution.

E. *Illustrative examples*

The following two example EMST problems are presented to illustrate the proposed ICA.

Example 1: The graph contains 6 vertices and 15 edges as shown in Fig. 10. The distance matrix is given in TABLE I. In Fig. 10, horizontal and vertical axes represent x and y coordinates of vertices respectively. This small size EMST problem is solved by ICA and the solution is plotted in Fig. 11.

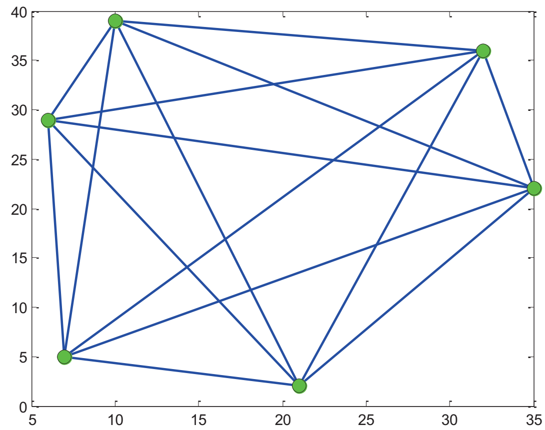


Fig. 10. Graph of Example 1

TABLE I
DISTANCE MATRIX OF EXAMPLE I

Node	1	2	3	4	5	6
1	0	24.02	26.93	30.89	29.83	10.77
2	24.02	0	39.82	14.32	32.76	34.13
3	26.93	39.82	0	35.74	14.32	22.20
4	30.89	14.32	35.74	0	24.41	38.60
5	29.83	32.76	14.32	24.41	0	30.23
6	10.77	34.13	22.20	38.60	30.23	0

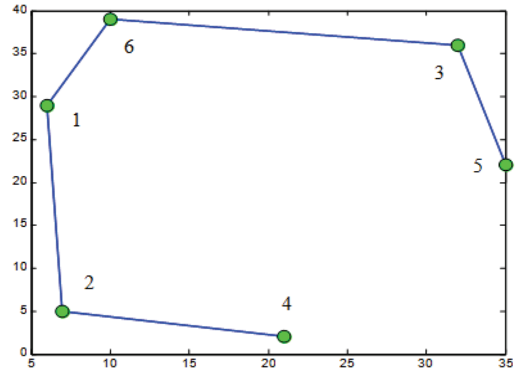


Fig. 11. Minimum spanning tree obtained by ICA for Example 1

Example 2: The graph contains 15 vertices and 105 edges. Its distance matrix is given in TABLE II. The solution obtained by ICA is plot in Fig.12 and its convergence is illustrated in Fig. 13. ICA took 18.27 seconds and 630 iterations to converge.

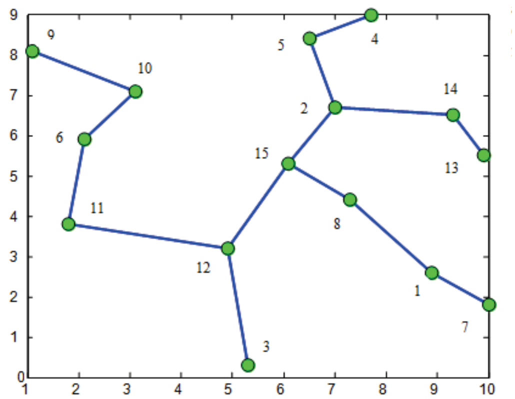


Fig. 12. Minimal spanning tree obtained by ICA for Example 2

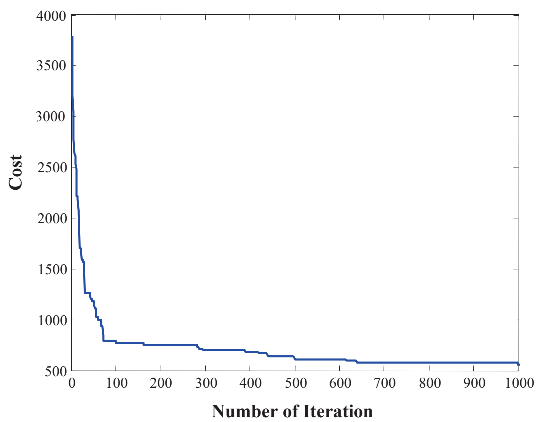


Fig. 13. Convergence diagram for Example 2.

TABLE II
DISTANCE MATRIX OF EXAMPLE 2

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	45.19	42.72	65.12	62.77	75.58	13.60	24.08	95.44	73.41	72.01	40.45	30.68	39.20	38.90
2	45.19	0	66.22	24.04	17.72	49.65	57.45	23.19	60.64	39.20	59.54	40.82	31.38	23.09	16.64
3	42.72	66.22	0	90.25	81.88	64.50	49.34	45.62	88.59	71.47	49.50	29.27	69.43	73.78	50.64
4	65.12	24.04	90.25	0	13.42	64.01	75.58	46.17	66.61	49.77	78.64	64.40	41.34	29.68	40.31
5	62.77	17.72	81.88	13.42	0	50.61	74.71	40.79	54.08	36.40	65.76	54.41	44.69	33.84	31.26
6	75.58	49.65	64.50	64.01	50.61	0	89.01	54.12	24.17	15.62	21.21	38.90	78.10	72.25	40.45
7	13.60	57.45	49.34	75.58	74.71	89.01	0	37.48	109.04	87.01	84.40	52.89	37.01	47.52	52.40
8	24.08	23.19	45.62	46.17	40.79	54.12	37.48	0	72.20	49.93	55.33	26.83	28.23	29.00	15.00
9	95.44	60.64	88.59	66.61	54.08	24.17	109.04	72.20	0	22.36	43.57	62.01	91.76	83.55	57.31
10	73.41	39.20	71.47	49.77	36.40	15.62	87.01	49.93	22.36	0	35.47	42.95	69.86	62.29	34.99
11	72.01	59.54	49.50	78.64	65.76	21.21	84.40	55.33	43.57	35.47	0	31.58	82.76	79.71	45.54
12	40.45	40.82	29.27	64.40	54.41	38.90	52.89	26.83	62.01	42.95	31.58	0	55.04	55.00	24.19
13	30.68	31.38	69.43	41.34	44.69	78.10	37.01	28.23	91.76	69.86	82.76	55.04	0	11.66	38.05
14	39.20	23.09	73.78	29.68	33.84	72.25	47.52	29.00	83.55	62.29	79.71	55.00	11.66	0	34.18
15	38.90	16.64	50.64	40.31	31.26	40.45	52.40	15.00	57.31	34.99	45.54	24.19	38.05	34.18	0

Both examples are also solved by LINGO, a commercial solver. TABLE III summarizes the performance of ICA against LINGO. Both algorithms reached the same minimum spanning tree but ICA took much less computational time.

TABLE III
RESULTS FROM ICA AND LINGO

Graph	Size		Solution Cost		Run Time (Second)	
	v	m	ICA	LINGO	ICA	LINGO
Example 1	6	15	85.63	85.63	1.81	2.17
Example 2	15	105	279.44	279.44	7.85	18.62

IV. CONCLUSION

This paper applies ICA, a new meta-heuristic algorithm, to the EMST problem. The efficiency of the proposed algorithm was demonstrated by two illustrative examples. Both of the examples supports that the proposed algorithm is fast and yields quality solutions. The following recommendations are given for future research.

1) Implementing ICA to other minimum spanning tree problems such as degree constraint minimum spanning tree (DMST) and capacitated minimum spanning tree (CMST) problems. Both are NP-hard problems.

2) Developing a hybrid technique to DMST and CMST problems that considers the combination of two or more meta-heuristics. For example, ICA may be integrated with GA, Tabu Search, or ACO.

REFERENCES

[1] J. Nesetril, "Otakar Boruvka on minimum spanning tree problem Translation of both the 1926 papers, comments, history," *Discrete Math.*, vol. 233, pp. 3–36, 2001.

[2] C. F. Bazlamacci and K. S. Hindi, "Minimum-weight spanning tree algorithms: A survey and empirical study," *Computers Oper Res,* vol. 28, no. 8, pp. 767–785, 2001.

[3] F. Neumann and C. Witt, "Ant colony optimization and the minimum spanning tree problem," *Theoretical Comput Sci*, vol. 411, no. 25, pp. 2406-2413, 2010.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms*, Cambridge, MA: MIT Press, 2009, pp. 631–642.

[5] J. Bentley and J. Friedman, "Fast Algorithms for Constructing Minimal Spanning Trees in Coordinate Spaces," *IEEE Trans. Comput.*, vol. 27, pp. 97–105, 1978.

[6] A. Yao, "On constructing minimum spanning trees in *k*-dimensional spaces and related problems," *SIAM J. Comput.*, vol. 11, no. 4, pp. 721–736, 1982.

[7] P. K. Agarwal, H. Edelsbrunner, O. Schwarzkopf and E. Welzl, "Euclidean minimum spanning trees and bichromatic closest pairs," *Discrete Comput. Geom.*, vol. 6, pp. 407–422, 1991.

[8] W. B. March, P. Ram and A. G. Gray, "Fast Euclidean minimum spanning tree: algorithm, analysis, and applications," in *Proc. 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, *KDD '10*, New York, NY, pp. 603–612, 2010.

[9] E. A-Gargari and C. Lucas, "Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition," in *IEEE Congress on Evolutionary Computation*, *CEC'07,* Singapore, pp. 4661-4667, 2007.

[10] J. Behnamian and M. Zandieh, "A discrete colonial competitive algorithm for hybrid flowshop scheduling to minimize earliness and quadratic tardiness penalties," *Expert Syst. Appl.,* vol. 38, no. 12, pp. 14490-14498, 2011.

[11] L. K. Shahvandi, M. Teshnehlab and A. Haroonabadi, "Multi-level clustering by imperialist competitive algorithm in wireless sensor networks," *International J. Adv. Eng. Sci. Technol.,* vol. 9, no. 2, pp. 327-332, 2011.

[12] S. Moadi, A. S. Mohaymany and M. Babaei, "Application of imperialist competitive algorithm to the emergency medical services location problem," *International J. Artif. Intell. Appl.*, vol. 2, no. 4, 2011.