A Survey of Two Optimization Methods to Solve a Modified Minimal Spanning Tree Problem in Undirected Tree Graph

Isra N. Alkallak
israalkallak19@gmail.com
College of Nursing
University of Mosul, Iraq

Ruqaya Z. Sha'ban
nnabeel2013@gmail.com
College of Medicine
University of Mosul, Iraq

Received on: 19 / 7 / 2010 **Accepted on: 16 / 3 / 2011**

ABSTRACT

The paper tackled a survey of two optimization methods to study spanning tree problem by modifying the spanning tree problem to generate all of possible solutions in undirected tree graph with simulated annealing algorithm and ant colony optimization algorithm. These algorithms are two of the optimization methods to find optimal solution from many of solutions in search space. A program is written in MATLAB 6.5 language to simulate these two algorithms with spanning tree problem. The experimental results in this paper show the effectiveness and easy implementation of each algorithm to find optimal solution, and to perform significantly better than the manual method.

Keywords: simulated annealing, ant colony, spanning tree.

استعراض طريقتين أمثليتين لحل مسألة الربط الشجري الأصغر المطورة في البيان الشجري غير الموجه

اسراء نذير الكلاك
كلية التمريض، جامعة الموصل

رقية زيدان شعبان
كلية الطب، جامعة الموصل

تاريخ استلام البحث: 2010/07/19 **تاريخ قبول البحث: 2011/03/16**

الملخص

استهدف هذا البحث استعراض طريقتين من طرائق الأمثلية، لدراسة مسألة الربط الشجري من خلال إضفاء بعض التطوير على مسألة الربط الشجري لتوليد جميع الحلول الممكنة في البيان الشجري غير الموجه، مع كلٍ من خوارزميتي محاكاة انصهار الصلب في المعادن، ومستعمرة النمل المثلى واللتان تعدان من مسائل الأمثلية، لانتقاء الحل الأمثل من بين العديد من الحلول. تم إعداد برنامج حاسوبي بلغة ماتلاب 6.5 ليحاكي كلتا الخوارزميتين مع المسألة. أثبتت نتائج البحث سهولة مرنة وكفاءة عالية لكل خوارزمية في انتقاء الحل الأمثل من بين العديد من الحلول في فضاء بحث المسألة عن استخدام الطرائق اليدوية في إيجاد حل المسألة.

الكلمات المفتاحية: انصهار الصلب، مستعمرة النمل، الربط الشجري.

1. Introduction

A network consists of a set of nodes linked by edges (or branches). The notation for describing a network is (V, E), where V is the set of nodes, and E is the set of edges between pairs of nodes. An edge is said to be directed or oriented if it allows positive flow in one direction and zero flow in the opposite direction. Minimal spanning tree algorithm is one of the best known problems in combinatorial optimization. Minimal spanning tree is a connected and undirected graph with weighted edges; a minimal spanning tree of the graph is a least-weight tree connecting all nodes [23]. For the minimal spanning tree problems, each edge is assigned a length. The goal of the problem is to select a spanning tree so that the total of the lengths of the selected edges is minimized. Given a connected undirected graph G = (V,E) and a weight w (u, v) specifying weight of the edge (u, v) for each edge (u, v). If $(v_i , v_j) \in$ V, then the two nodes v_i & v_j are said to be adjacent in G, edge (v_i , v_j) is then said to be incident to nodes v_i and v_j and v_i is a neighbor of v_j. A loop is an edge whose endpoints are equal [13].

A path in G is an order list of distinct nodes $(v_1,v_2,\ldots v_{q-1},v_q) \in$ V, if $v_1= v_q$ a closed path containing at least one edge is called a cycle. G is a tree, if G is connected and has no cycle. H = (v, e) is called subgraph of G, if $v \subset$ V and $e \subset$ E. A subgraph H = (v, e) of G is called a spanning subgraph of G if H contains all the nodes v of G. A spanning

tree is a connected network that may involve only a subset of all the nodes of the network with no cycle allowed as illustration in Figure (1), [18]. Thus, a spanning tree is a subgraph of a graph G, which contains all nodes from G with no cycle. The minimum spanning tree of a weighted graph is a minimum weight spanning tree of that graph [8, 13, 18, 22, 23]. The network with E nodes, spanning tree is a group of E-1 edges that connects all nodes of the network and contains no loops [22].

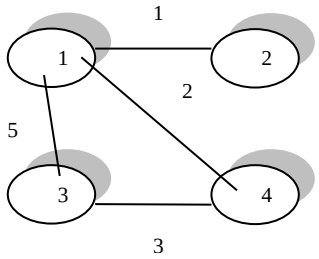


Figure (1). A weighted tree graph

The ground states of a complex physical system such as a solid can be reached by heating the system up to some high temperature and then cooling it down slowly. This process is called annealing. Annealing method is simulated to solve combinatorial optimization problems [10,16].

The ant colony optimization metaheuristic is also a technique for solving combinatorial optimization problems. The origin of ant algorithms is to imitate the behavior of ants searching for food. Ants are finding sources of food in the several ways as they explore the area surrounding their nest in a random manner. While they are moving, the ants left a pheromone (chemical trace) on the floor, in such a way that they can find their way back to the nest. When they find a source of food, the ants bring food back to the nest following the pheromone trace [19]. Therefore, a heuristic method is designed toward promising regions of search space containing high quality solutions [21].

In simulated annealing, a single agent is deployed for a single beam session, while ant colony optimization uses multiple agents, each of which has its individual decision made based upon collective memory or knowledge [21].

The aim of this paper is to find the set of edges connecting all nodes such that the sum of the edge lengths from any node to the last one in the graph is minimized by the proposed additional steps (modified spanning tree algorithm) i.e., visiting all neighbor solutions in the search space. Two methods of heuristic namely; simulated annealing and ant colony optimization are summarized and examined to solve the combinatorial optimization problems as a minimal spanning tree problem.

In this paper, besides this introductory section, section 2 presents the simulated annealing algorithm. The ant colony optimization algorithm is described in section 3. section 4 contains the modified procedures that are carried on the two prementioned algorithms with the steps added for more accurate in passing local minima. Experimental results of the modified spanning tree algorithm are examined in section 5. This section also contains some valuable discussions. Finally, section 6 concludes this paper.

2. Simulated Annealing Algorithm

In 1953, Metropolis *et al*. developed a simulation procedure based on the Monte Carlo method. In 1983, Kirkpatrik *et al*. developed that algorithm after the process of cooling glass from a high temperature to a low one, known as annealing. The advanced

simulated annealing algorithm is therefore an algorithm which simulates the annealing process with Metropolis Monte Carlo simulation as a probabilistic acceptance rule. Metropolis step allows the system to move consistently toward lower energy states; yet still jumps out of local minima probabilistically, when the temperature decreases logarithmically [23].

Simulation annealing algorithm is a non-traditional method possessing both greedy (deterministic) and random (stochastic) characteristics [15]. The deterministic aspect attempts to improve upon the current state using a predefined cost function. However, the stochastic aspect occasionally accepts a state that is not an improvement [6]. Simulated annealing is a global heuristic technique which tries to avoid falling into local optima by accepting bad solutions when specific function conditions are satisfied [12]. It has been considered as a good tool for complex nonlinear optimization problems. The technique has been widely applied to a variety of problems. One of the major drawback of the technique is its very slow convergence. Often the solution space of an optimization problem has many local minima. However, in such optimization algorithm, a simple local search algorithm proceeds by choosing random initial solution and generating a neighbor from that solution. The neighboring solution is accepted if it is a cost decreasing transition. The simulated annealing algorithm, though by itself it is a local search algorithm, avoids getting trapped in a local minimum by accepting other cost increasing neighbors with some probability. In simulated annealing, first an initial solution is randomly generated, and a neighbor is found [5].

The simulated annealing procedure simulate the process of slow cooling of molten metal to achieve the minimum function value in a minimization problem. It is a point –by-point method. The algorithm begins with an initial point and a high temperature T. A second point is taken at random in the vicinity of the initial point and the difference in the function values (ΔE) at these two points is calculated. The second point is chosen according to the Metropolis algorithm which states that if the second point has a smaller function value, the point is accepted; otherwise the point is accepted with a probability exp (-(ΔE / T). This completes one iteration of the simulated annealing procedure.

In the next generation, another point is created at random in the neighborhood of the current point and the Metropolis algorithm is used to accept or reject such point. In order to simulate the thermal equilibrium at every temperature, a number of points is usually tested at a particular temperature before reducing the temperature. The temperature is then reduced according to a temperature schedule called simulated annealing schedule (or annealing schedule in short). The term annealing comes from the technique of hardening a metal (i.e. finding a state of its crystalline lattice that is highly packed) by hammering it, while being initially very hot and then at a succession of decreasing temperatures [7]. The algorithm is terminated when a sufficiently small temperature is obtained or a small enough change in the function values is found [8,10,11,15,16,20].

2.1 Cooling Schedules

It is important that the cooling function allows sufficient time to explore many possible solutions in one level before moving into lower temperature and ultimately freezing point. It is also important not to spend too much time on high temperatures, where most neighbourhood moves are accepted as this can lead to a wastage of running time. A good schedule is expected to spend more time on lower temperatures so as to allow for convergence. It is not advisable to spend too much time on low temperatures

where most neighbourhood moves are rejected [3]. Starting at a high value of the initial control parameter and then decreasing it at a specified rate after completion of solution by the rule [12].

$$T_k = \alpha T_{k-1}$$

where $(0 < \alpha < 1)$ α is a scaling factor and T_{k-1} & T_k are previous and forward temperatures, respectively. The decreased rate is depended on the problem solving.

2.2 A Generic Description of the Simulated Annealing Algorithm [10]

1. Get an initial state with energy x (solution).
2. Make this initial state be the current state.
3. Select an initial "high temperature" T.
4. While the system is "not yet frozen" do

```
Begin
While the system is "not yet in thermal equilibrium" do
     Begin:
      Pick a random "nearby" state with energy xp .
      Let Δx = xp - x.
      If Δx <= 0
         The newly proposed state becomes the current state,
      Else
          The newly proposed state becomes the current state
             With probability = e -Δx / T
          Else no change in state (i.e., reject state).
    End
  "Reduce the temperature T by ΔT".
   End
```

5. Output the current state.

3. Ant Colony Optimization Algorithm

Ant colony optimization algorithms are population based optimization approaches that have been applied to solve different combinatorial optimization problems. The inspiring source of ant colony optimization is the foraging behavior of real ants. When searching for food, ants initially explore the area surrounding their nest in a random manner. As soon as an ant finds a food source, it evaluates the quantity and the quality of the food and carries some of it back to the nest. During the return trip, the ant deposits a chemical pheromone trail on the ground. The quantity of pheromone deposited, which may depend on the quantity and quality of the food, will guide other ants to the food source. Indirect communication between the ants via pheromone trails enables them to find shortest paths between their nest and food source. This characteristic of real ant colonies is exploited in artificial ant colonies in order to solve combinatorial optimization problems. Ant colony optimization algorithms update the pheromone values using previously generated solutions. The update aims to concentrate the search in regions of the search space containing high quality solutions [1, 17, 19,21].

Ant colony optimization is metaheuristic method. Artificial ants are characterized as agents that imitate the behavior of real ants. However, it should be noted that an artificial ant colony system has some differences in comparison with a real Ant colony system, as follows

- Artificial ants have memory.
- They are not completely blind.
- They live in an environment where time is discrete.

On the other hand, an artificial Ant colony system has several characteristics adopted from real ant colony system:

- Artificial ants have a probabilistic preference for paths with a large amount of pheromone.
- Shorter paths tend to have larger rates of growth in their amount of pheromone.
- The ants use an indirect communication system based on the amount of the pheromone deposited in each path [2, 21].

The key idea is that, when a given ant has to choose between two or more paths. The path that was more frequently chosen by other ants in the past will have a greater probability of being chosen by the ant. Therefore, trails with greater amount of pheromone are synonyms of shorter paths. Ant colony system performs a loop containing two basic procedures, namely:

- A procedure specifying how the ants modify solutions of the problem being solved in a probabilistic way. The probability of adding a new item to the current partial solution is given by a function that depends on problem and on the amount of pheromone deposited by ants on this trail in the past.
- A procedure to update the pheromone trails are implemented as a function that depends on the rate of pheromone evaporation and on the quality of the produced solution [21].

In this paper, each ant in the colony chooses an arbitrary edge depended on the proposed additional steps with the problem constraints. The procedure continues for each ant in the colony has chosen an edge. Finally a set amount of pheromone is added to that chosen edge, and the ant in the colony chooses another edge. The amount of pheromone added to an edge is obtained by optimal solution

3.1 Pheromone Update

Real ants deposit a substance called pheromone while moving from one point to another point. Artificial ants perform this action by adding a value called trace on the trail levels of moves chosen by them. The updating of trail levels can be performed either after each move or after completion of solution. In 1996, Dorigo *et al*. have experimentally shown that the performance of second way is much better than the first [17].

In this paper, each edge in the graph is given an initial pheromone value *ph* equal to 1. There are *n* nodes in the graph, (assuming that the size of the colony of ant is *n*). Each ant will start its tour from a different node. When all ants finish their tour, they will track back and update the pheromone along their path by putting additional pheromone Δph. For each edge, the new pheromone value is as follows:

$$ph = ph + \Delta ph$$

with $\Delta ph = L / S$

where *L* is the summation of all edge's weight in the graph. *S* is the length of the path built by the ant.

The pheromone evaporation (*p*)is defined is that factor which enables greater exploration of the search space and minimizes the chance of premature convergence to suboptimal solution by all ants in the colony. Colony optimization algorithm uses an

evaporation rate in order to forget previous bad choices at the cost of loosing useful information. Pheromone evaporation can help the ants escape from local minimum. Where $0 < p < 1$ [4].
The pheromone will evaporate is as follows:

$ph = (1- p) * ph$

In this paper, it is assumed that p is equal to 0.2.

The heuristic information indicates the desirability of assigning the object *i* to the location *j*. There are several methods to estimate the desirability depending on the problem. In this paper, the artificial ants can move from one node in the tree to another without including heuristic information. That because the ants move in some deterministic way and they visit all edges in the search space depending on the constraints of problem. In 1997, Taillard and Gambardella proposed a fast ant colony algorithm namely FANT for quadratic assignment problem. In FANT algorithm, each iteration uses only one ant and assignments are made without heuristic information [17]. Also in this paper, one of attributes of FANT algorithm is used without heuristic information.

3.2 Termination Condition

Ant colony optimization algorithm is similar to other metaheuristic such as simulated annealing. The ant algorithm can be terminated in several manners, e.g. repeating the algorithm for a maximum number of iterations or running for a stipulated time. Some researchers run the algorithm and the best objective function value obtained in each iteration is recorded and compared with that obtained in earlier iterations. The algorithm stops if the objective function value has not been improved within the last certain number of iterations [17].

In this paper, termination manner belongs to the way that run the algorithm and the best objective function value obtained in each iteration is recorded and compared with that obtained in earlier iterations. The algorithm stops by the number of iterations.

4. The Modified Procedure

4.1 Preliminaries

The minimal spanning tree problem was originally stated by Boruvka in 1926. In many references, several algorithms are noticed to solve the minimal spanning tree problem, such as Prim's algorithm which repeatedly adds edge (u,v) of minimum weight such that $u \in V_{new}$ (V_{new} is selected vertex) and $v \in V - V_{new}$, and add v to V_{new} until $V= V_{new}$ [14]. Another algorithm as Kruskal's algorithm sorts edges and then adds edge of minimal weight first such that no cycles are created [9]. These algorithms are examples of the minimal spanning tree algorithms which are often referred to as Greedy algorithms. Such algorithms do not yield globally optimal solutions [22].

Additional steps are proposed here to the spanning tree algorithm to create many neighbor states (neighbor solutions) in graph. These states will have different weights (different solutions). The uncontinuing Greedy algorithm yields globally optimal solutions. Just like Prim's and Kruskal's algorithms, these additional steps are based on the minimal spanning tree algorithm. Another reason; a graph often contains redundancy in that there can be multiple paths between some or among all nodes. Therefore, the proposed additional steps (modified spanning tree algorithm) visit all neighbor solutions in the search space. These steps may force us later to follow a "bad

edge" to explorer all probabilities solutions in graph tree (neighbors in search space). A survey of two algorithms (simulated annealing algorithm, ant colony optimization) is given here and can reach optimal solution.

4.2 The Proposed Steps

Figure (2-a) tree graph illustrates the following proposed steps:

Step 1

Input the connected network with n of nodes
Let C = 0, solution = 0, C' = n

Step 2

Choose a short weight of edge between any nodes in connected network
that represent as edge (i,j).

Step 3

Assigned the two nodes i, j from connected set in C as C = {i, j}.
The remaining nodes in the network (unconnected set of nodes) are assigned in C' as C' = {q,m}.

Step 4

For each node in C do following:
{ if we select the node i then the node j is the closest node to the node i, the two nodes i and j in C as C = {i, j}, and edge (i,j) will be in the minimum spanning tree}.

Step 5

Select arbitrarily, one of the node's successors in set C
{ the successors of nodes i, j is m, q}.
{ the edges are edge (i, m), edge (i, q), edge (j, m)}.

Step 6

Compute the solution as solution = solution + edge's weight
{for example solution = solution + edge's weight (i,m)}

Step 7

Update the C and C' by adding the new node m to C and deleting from it
C' as C = {i , j, m}.

Step 8

If C' = ø goto step 9 {Test the set C'}
Else
Repeat step 5

Step 9

Print the solution and the path of nodes in graph

Step 10

End

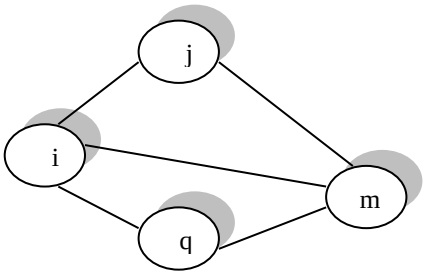


Figure (2-a). Tree graph

Figure (2-b) illustrates the proposed steps graphically. Figure (3) illustrates the flowchart of simulated annealing algorithm to solve spanning tree problem. Figure (4)

illustrates the flowchart of ant colony optimization algorithm to solve minimal spanning tree problem.

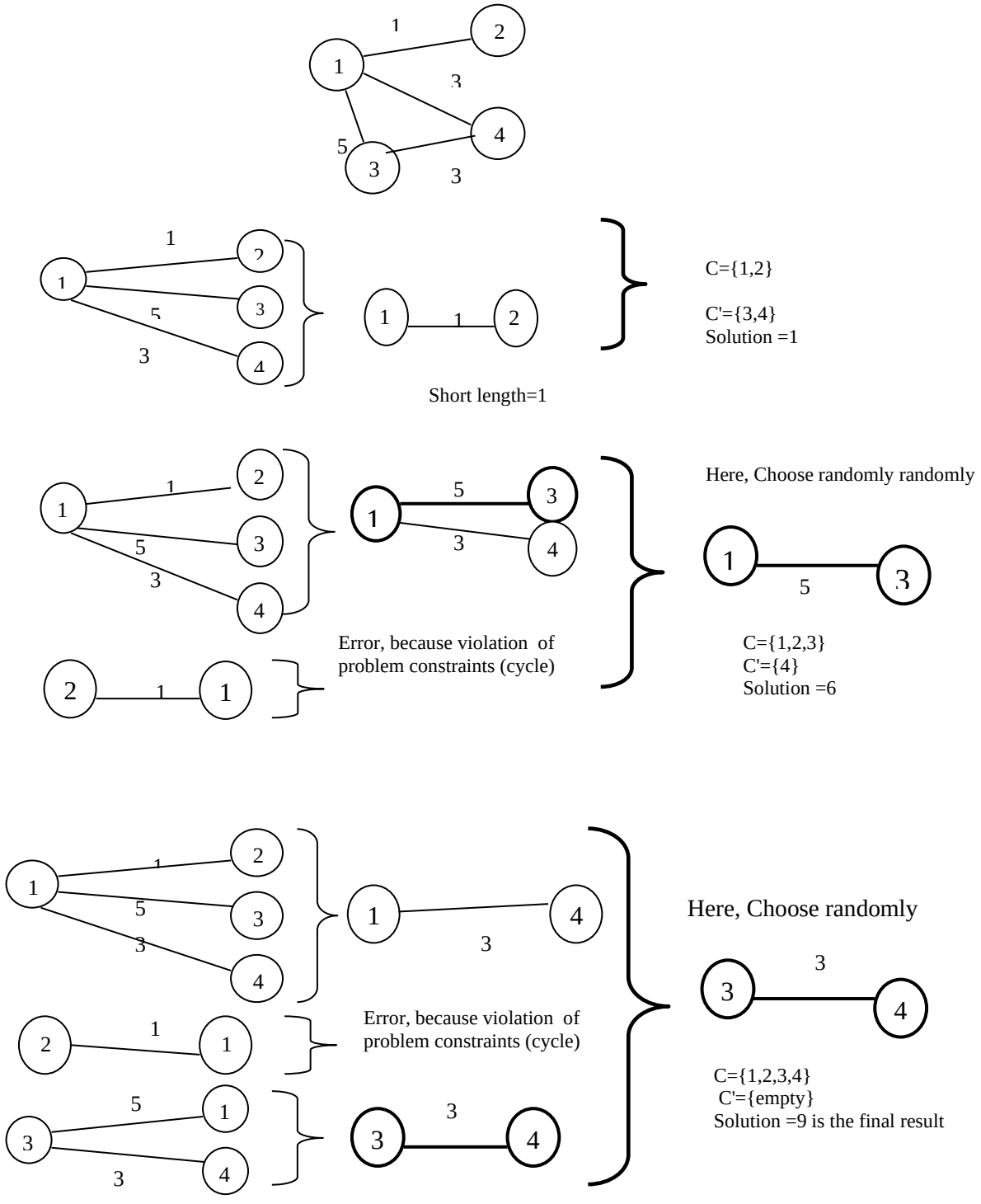


Figure (2-b). Proposed steps graphically

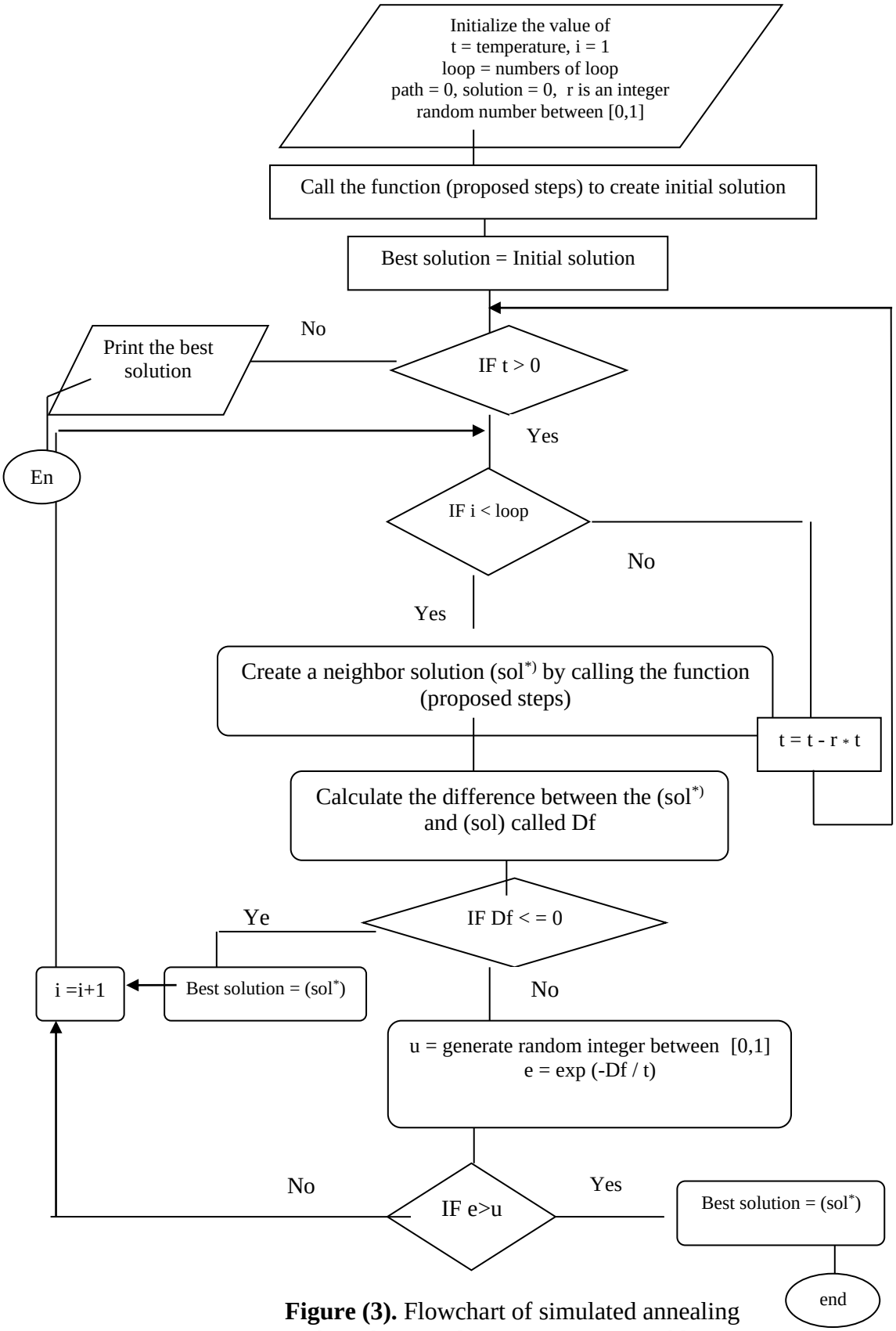


Figure (3). Flowchart of simulated annealing algorithm to solve spanning tree problem

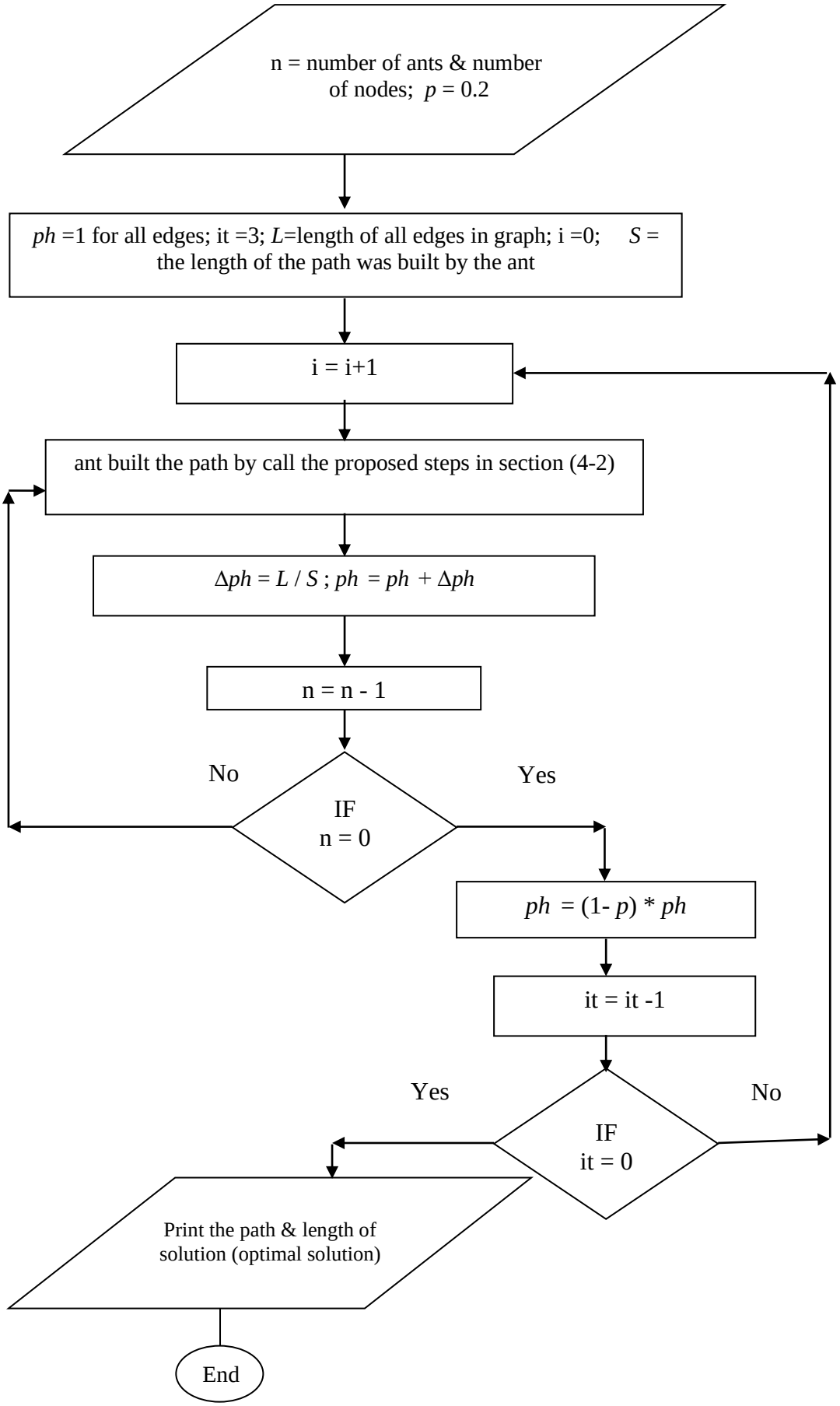
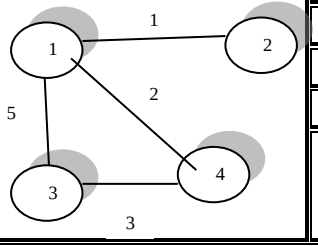


Figure (4). Flowchart of ant colony algorithm to solve minimal spanning tree problem

4.3 Search Space (Configuration) of Problem

The Search space is the space of all possible solutions that can be considered (visited) during the search. In this paper, the nodes, edges and edge's weights of search space in are shown in Table (1)

Table (1). Edges of tree graph

Tree Graph	Node number	Edge	Edge's Weight
	1	(1,2),(1,3),(1,4)	1,5,2
	2	(2,1)	1
	3	(3,1),(3,4)	5,3
	4	(4,1),(4,3)	2,3

4.4 Initial Solution (Configuration)

The initial configuration is specified at the beginning of the search in the search space. In this paper, the above steps in section (4-2) are used to generate the initial solution.

4.5 Neighborhood Structure

An immediate neighbor of the current trial solution is the one that is reached by adding a single link and then deleting one of the other links in the cycle that is formed by the addition of this link [20]. The neighborhoods contain all the feasible route configurations that can be obtained from the current solution. The neighborhood generally consists of solutions obtained by making small changes in the previous solution.

In this paper, a neighborhood configuration is constructed by making a single swap of current values of two arbitrary successors. The simulated annealing algorithm checks the objective function between the current solution and neighbor solution. The ant colony optimization algorithm is used also for the same purpose after evaluating the whole set of neighboring solutions and selecting the best of the objective functions.

4.6 Move Set for Spanning Tree

In Figure (5), number of states to generate neighbor solution are illustrated.

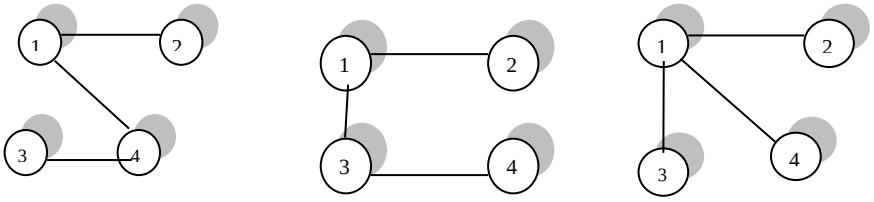


Figure (5). states of neighbor solution

4.7 ETIME Function

When implementing the simulated annealing algorithm, the algorithm is still able to converge with duration of time, depending on the ETIME Elapsed time function in MATLAB 6.5 language. ETIME (T1, T0) returns the time in seconds that has elapsed between vectors T1 and T0. The two vectors must be of six elements long, in the format returned by CLOCK given by:

T = [Year Month Day Hour Minute Second]

Time differences over many orders of magnitude are computed accurately. The result can be thousands of seconds if T1 and T0 differ in their first five components or small fractions of seconds if the first five components are equal.

```
T0 = clock;
Operation
T1=clock
ETIME (T1, T0)
```

The time in seconds needed to perform a simulated annealing algorithm with minimal spanning tree is illustrated in Table (2).

4.8 Objective Function

An optimization problem, of course, comes with an objective function to be minimized. In this paper, the energy function for minimal spanning tree include that there should be:

1. A subgraph of graph G, (spanning tree) which contains all nodes from G.
2. Spanning tree contains no cycle.
3. The weight of edges in spanning tree that is minimized.

Many attempts are faced to find a better solution. In implementation the simulated annealing algorithm has different temperature mapping causing too much vibration between current and next state to reach the solution. Since, differences between the next and current solution are small, probability of switching over becomes close to one, while temperature is either not growing or growing too slowly in the algorithm to reach the solution. The algorithm runs efficiently several times.

5. Experimental Results and Discussion

5-1. Results of Simulated Annealing Algorithm

A program is written in MATLAB 6.5 language. The core of the code programming to implement the simulated annealing algorithm consists of many functions, and the program makes use of other functions to perform the changes in the solution space. The input parameters of the program are the number of iterations, the search space, the matrix contains the number of the nodes, distances between these nodes (weights of the edges), control parameter or temperature decrease rate, and the stopping rule. The final output of the program is a table contains the value of temperature, value of loop, the best solution, time, path, current solution, old solution, difference between them, probability value and random value as illustrated in Table (2) and Appendix. The search is terminated after reaching the optimal solution, or after some number of iterations without an improvement in the objective function value.

Table (2). Some results of simulated annealing algorithm

Value of temperature	Value of loop	The best solution	Time	path
5	5	8	248.27770	[1 2,1 3,1 4]
5	7	8	92.7780	[1 2,1 4,1 3]
5	10	6	256.9790	[1 2,1 4,4 3]
10	5	8	157.7290	[1 2,1 3,1 4]
10	10	6	336.3050	[1 2,1 4,4 3]
10	50	6	718.2780	[1 2,1 4,4 3]
100	5	8	162.4170	[1 2,1 3,1 4]

100	50	9	1.0522e+003	[1 2,1 3,3 4]

Below some solutions in Table (2) as resulting solution is 8 and the path from beginning to the end of the solution is [1 2 ,1 3, 1 4] as illustrated graphically in Figure (6)

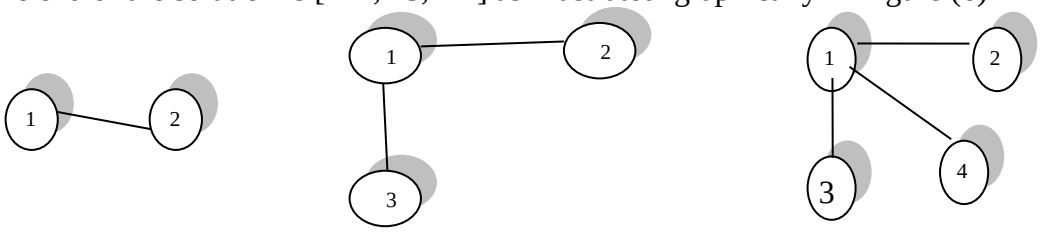


Figure (6). Solution of spanning tree

Another the resulting solution is 9 and the path from beginning to the end of the solution is [1 2 ,1 3, 3 4] as illustrated graphically in Figure (7)

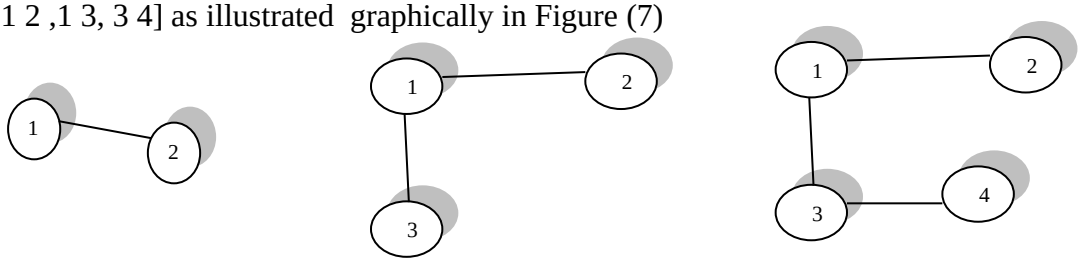


Figure (7). Solution of spanning tree

The resulting best solution is 6 and the path from beginning to the end of the best solution is [1 2 ,1 4, 4 3] as illustrated graphically in Figure (8)

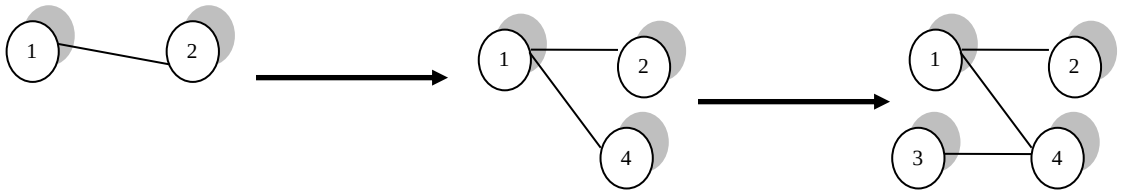


Figure (8). best solution of spanning tree

For example, the results in appendix illustrate the algorithm begins a high temperature t is 5. The number of iterations is 10. Call the proposed steps to create initial solution. Suppose the initial solution is best solution. Exam temperature is greater than zero and number of iterations is not equal zero to continue the algorithm. Call the proposed steps to create neighbor solution. Calculate the difference (Df) between neighbor solution and old solution. If difference less than or equal zero then neighbor solution is best solution otherwise generate random integer between [0,1] as u. Calculate exp (-(Df / t) by e. If e greater than u then neighbor solution is best solution otherwise increment iteration by 1. These complete one iteration of the simulated annealing procedure and continue for loop. Choose r is an integer random number between [0,1]. The temperature is then reduced according to the roll $t = t - r * t$. Temperature initially is very hot and then at a succession of decreasing temperatures. The algorithm is terminated when a sufficiently small temperature is obtained or a small enough change in the function values is found. Finally, the results in appendix illustrate the temperature near by zero and the best solution is 6.

5-2. Results of Ant Colony Optimization Algorithm

The code programming to implement the ant colony optimization algorithm that simulates this research consists also of many functions. The ant starts the graph traversal

from an arbitrary selected node. It traverses edges until all edges are finished. Initial parameters are loaded as iteration, initial pheromone value, number of ants (equal number of nodes in the graph). Each of the edges is set with an initial pheromone value of 1. Ants are individually placed on arbitrary nodes.

After a number of ants have traversed the graph known as one iteration of ant colony optimization, the amount of pheromone is also evaporated from all edges, pheromone evaporation occurs in nature and in ant colony optimization. It can help the ants escape from local minimal. When examining the ant colony optimization algorithm, it can be seen that ants will initially tend to freely explorer the whole solution space, leading to many different solutions, however, overtime, pheromone will accumulate only on edges that are part of the those traversals. The pheromones are set to random initial values. They pheromones are necessary for later iterations of the algorithm, when they reinforce the traversal of the graph edges that lead to a good solution.

When traversing from a node *i* to a node *j*, the probability of an ant *k* choosing the edge that connects the node *i* with node *j* is given by the additional steps in section (4-2), and put the pheromone for every traversed edge.

In this paper, after terminating of iterations and an artificial ant has finished constructing solutions, the pheromone is updated. The updating of pheromone depends on the solution; is it feasible or not? The values of pheromones are sorted by descending. The maximum value of pheromone is selected with the solution. This solution is the best. The rest pheromones are ignored. After all ants have constructed solutions, the solutions are compared to those obtained by other ants. The minimum value of solution is the optimal one. Since no ant using the edges, had minimum values of pheromone, there is no additional pheromone given. Pheromone evaporation reduces the intensity of pheromone values on these edges. This will make such edges less attractive for future ants. The algorithm will proceed until a stopping criteria is met.

Explanation of how ants construct the solution about tree graph of Figure (1), is as follows: There are 4 nodes, (assuming that the number of ant is 4 also), each ant will start tour from different nodes; for example, the first ant starts from node 1, the second ant starts from node 2, and so on.

Iteration 1

The first ant starts from node 1; there are three neighboring nodes to be visited. Any other ant also has three neighboring nodes to be visited. The probability of choosing any edges is calculated using the proposed steps in section (4-2) with the problem constraints. Table (3) illustrates the track of each ant in the colony in iteration 1.Table (4) illustrates repeated edges, ant's number visiting the edges, numbers of repeated edges and the pheromone update after performing pheromone evaporation procedure in iteration 1.

Table (3). First track

ants	paths of each ant in graph	length of path (*S*)	$\Delta ph = L / S$
ant1	[1 2,1 4,4 3]	6	1.8333
ant2	[2 1,1 4,1 3]	8	1.3750
ant3	[3 1,1 4,1 2]	8	1.3750
ant4	[4 1, 4 3,1 2]	6	1.8333

Table (4). Calculation pheromone for first track

repeated edges	ant's number visiting the edges	numbers of repeated edges	new pheromone ph = (1- p) * ph
1 2	1,2,3,4	4	5.9333
1 4	1,2,3,4	4	5.9333
4 3	1,4	2	3.7333
1 3	2,3	2	3

The resulting best solution is 6 and the path from beginning to the end of the best solution is [1 4 ,1 2, 3 4] as illustrated graphically in Figure (9)

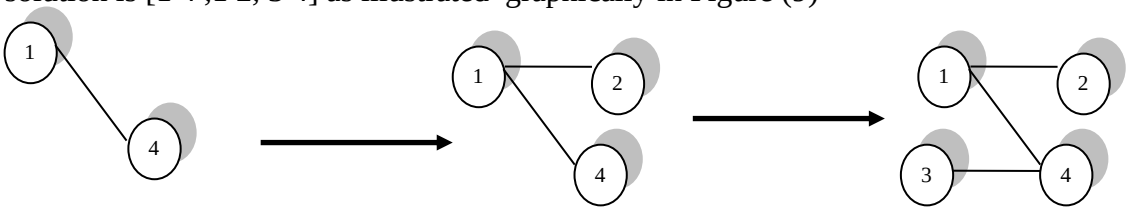


Figure (9). Solution of iteration 1

Iteration 2

Also in iteration 2, the same process is performed and repeated as in the first iteration. The initial pheromone values on all edges are changed. Table (5) illustrates the track of each ant in the colony, Table (6) illustrates repeated edges, ant's number visiting the edges, numbers of repeated edges and the pheromone update after pheromone evaporation procedure is performed in iteration 2.

Table (5). Second track

ants	paths of each ant in graph	length of path (S)	Δph = L / S
ant1	[1 2,1 3,1 4]	8	1.3750
ant2	[2 1,1 3,3 4]	9	1.2222
ant3	[3 1,3 4,1 2]	9	1.2222
ant4	[4 1,4 3,1 2]	6	1.8333

Table (6). Calculation pheromone for second track

repeated edges	ant's number visiting the edges	numbers of repeated edges	new pheromone ph = (1- p) * ph
1 2	1,2,3,4	4	9.2689
1 4	1,4	2	7.3133
3 4	2,3,4	3	6.4089
1 3	1,2,3	3	5.4556

In Iteration 2, the best solution is 6 and the path from beginning to the end of the best solution is [1 2, 1 4, 3 4] as illustrated graphically in Figure (10).

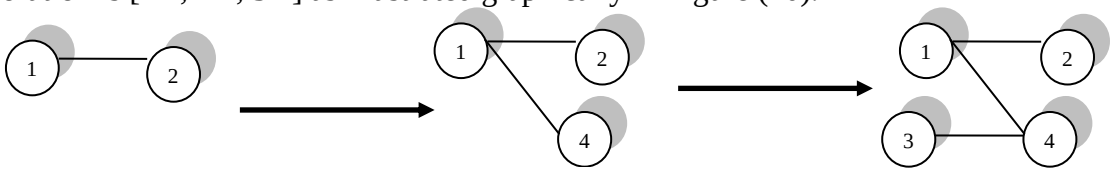


Figure (10). Solution of iteration 2

Iteration 3

The same process performed in the first iteration is repeated here. The initial pheromone values on all edges are modified. Table (7) illustrates the track of each ant in the colony, Table (8) illustrates the repeated edges, ant's number visiting the edges, numbers of repeated edges and the pheromone update after pheromone evaporation procedure is performed in iteration 3.

Table (7). Third track

ants	paths of each ant in graph	length of path (S)	Δph = L / S
ant1	[1 2,1 3,1 4]	8	1.3750
ant2	[2 1,1 4,4 3]	6	1.8333
ant3	[3 1,1 2,1 4]	8	1.3750

Table (8). Calculation pheromone for third track

repeated edges	ant's number visiting the edges	numbers of repeated edges	new pheromone ph = (1- p) * ph
1 2	1,2,3,4	4	12.1818
1 4	1,2,3	3	10.6173
1 3	1,3,4	3	7.6644

ant4	[4 1,1 3,1 2]	8	1.3750	4 3	2	1	6.5938

Here, the best solution is 8 and the path from beginning to the end of the best solution is [1 2,1 4,1 3] as illustrated graphically in Figure (11).

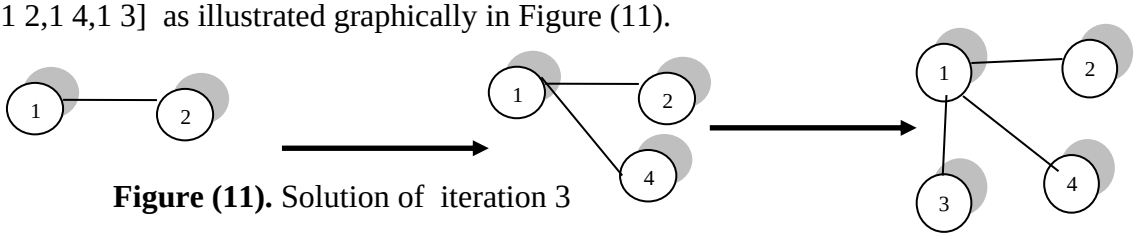


Figure (11). Solution of iteration 3

The search is terminated after reaching the optimal solution or after repeating some number of iterations without an improvement in the objective function value. In our example, the optimal solution (minimum value of length of path) is 6.

6. Conclusions

Simulated annealing algorithm and ant colony optimization are well suited for solving combinatorial optimization problems. Simulated annealing has proven an effective technique for problems where there are many solutions and no efficient way to derive which one is the best. A local solution is a state that is not the best solution but where any change from the current state update the cost. Ant colony optimization algorithm is easy to implement and is more efficient than the simulated annealing algorithm in computation time. The proposed steps perform significantly better than manual method. Simulated annealing and ant colony optimization perform better than any local optimization method and yield a solution close to global optimum. Despite the classical method (manual method) can give the exact solution, it takes a large space and needs more computations. Therefore, the proposed solution facilitates the difficultly in solving those problems that are proportional to the size, as the problem space and the number of nodes increased. The proposed solution will be close to be the global optimum. Moreover, our results show the advantages of this new approach over more traditional (manual approach).

REFERENCES

[1] Dorigo, M. and Blum, CH., 2005, Ant Colony Optimization Theory: A computer science, Elsevier, 344.pp. 243-278. www.elsevier.com/locate/tcs

[2] Dorigo, M., Maniezzo, V. and Colorni A., 1996, The Ant System: Optimization by a Colony of Cooperating Agents, IEEE, Vol.26, No.1, pp.1-13.

[3] Eglese R., W., 1990, Simulated Annealing: A Tool for Operational Research, European Journal of Operational Research, Vol. 46, pp.271- 281.

[4] Jalali, M.R., Afshar, A. and Marino, M.A., 2006, Improved Ant Colony Optimization Algorithm for Reservoir Operation, Scientia Iranica, vol.13, No.3, pp. 295-302.

[5] Janaki Ram, D., Sreenivas, T. H. and Subramaniam, G. K., 1996, Parallel Simulated Annealing Algorithms, Journal of Parallel and Distributed Computing, Vol.37,No.0121, pp.207-212.

[6] Jansen, M. B. J., Pooch, U., 1997, An Information Retrieval Application for simulated annealing, The 2nd ACM Conference on Digital Libraries. Philadelphia, PA., 259-260.

[7] Kirkpatrick, S. C. D., Gelatt, Jr and Vecchi M. P. 1983, Optimization By Simulated Annealing, Science, Vol. 220, No. 4598, pp. 671-680.

[8] Knowles, J. and Corne, D., 2000, A New Evolutionary Approach to the Degree-Constrained Minimum Spanning Tree Problem, IEEE Transactions on Evolutionary Computation, Vol. 4, No. 2.

[9] Kruskal, J. B., 1956, On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. Proc, Am. Math. Soc. 7(1), p 48-50.

[10] kwok-ching Lam, J., 1989, An Efficient Simulated Annealing Schedule, Ph.D., Computer Science, Yale University.

[11] Misevicius, A, Blazauskas, T., Blonskis, J., smolinskas, J., 2004, A Overview of Some Heuristic Algorithms for Combinatorial Optimization Problems, Informacines Technologijos. Ir Valdymas, No.1 (3).

[12] Mushi, A. R., 2007, Simulared Annealing Algorithm for the Examinations Timetabling Problem, African Journal of Science and Technology (AJST), Vol. 8, No. 2, pp. 24-32.

[13] Patwary, M. A., 2006, Hardness of Finding Minimum Face-Spanning Subgraphs of Plane Graphs, M.Sc., Computer Sciences and Engineering, Bangladesh University.

[14] Prim, R. C., 1957, Shortest Connection Networks and Some Generalizations. Bell Systems Technical JRnl. p1389-1410

[15] Sahu, A., Tapadar, R., 2007, Solving the Assignment Problem Using Genetic Algorithm and Simulated Annealing, IAENG International Journal of Applied Mathematics, 36:1, IJAM_36_1_7.

[16] Sharma, S. K. and Lees, B. G., 2004, A Comparison of Simulated Annealing and Gis Based Mola for Solving the Problem of Multi-objective Land Use Assessment and Allocation, MCDM, whistler, B.C. Canada. August 6-11.

[17] Solimanpur, M., Vrat, P. and Shankar, R., 2004, Ant Colony Optimization Algorithm to the Inter-Cell Layout Problem in Cellular Manufacturing, European Journal of Operational Research 157, pp.592-606. www.elsevier.com/locate/dsw

[18] Taha, H. A., 2003, Operation Research: An introduction, seventh, edition, Prentice Hall, pearson Education, International, Inc.

[19] Taillard, E.D., Gambardella, L. M., 1997, Adaptive memories for the quadratic assignment problem, Technical Report IDSIA-87-97, IDSIA, Lugano, Switzerland.

[20] Thamilselvan, R. and Balasubramanie, P., 2010, Integration of Metaheuristic Algorithms for Minimum Spanning Tree, (IJCNS)International Journal of Computer and Network Security,Vol.2, No. 2

[21] Thangavel, K, Karnan, M., Jeganathan, P., Petha Lakshmi, A., Sivakumar, R. and Geetharamani, G., 2006, Ant Colony Algorithms in Diverse Combinational Optimization Problems – A Survey, ACSE Journal, Vol. 6, No. 1.

[22] Winston, W. L., 1994, Operation Research Applications and Algorithms, Third Edition, printed in the United States of America wadsworth, Inc.

[23] Zhang, J., and Zhou, J., 2006, Models and Hybrid Algorithms for Inverse Minimum Spanning Tree Problem with Stochastic Edge Weights, World Journal of Modeling and Simulation, England, UK, Vol. 2, No. 5, pp. 297-311.

Appendix

The results of simulated annealing algorithm about example in figure (1):

Tempe.	Loop	current sol	old_sol	deferens	prop	rand	best_sol
5.0000	1.0000	9.0000	9.0000	0	0	0	9.0000
5.0000	2.0000	9.0000	9.0000	0	0	0	9.0000
5.0000	3.0000	6.0000	9.0000	-3.0000	0	0	6.0000
5.0000	4.0000	8.0000	6.0000	2.0000	0.6703	0.6813	6.0000
5.0000	5.0000	6.0000	8.0000	-2.0000	0.6703	0.6813	6.0000
5.0000	6.0000	9.0000	6.0000	3.0000	0.5488	0.7857	6.0000
5.0000	7.0000	8.0000	9.0000	-1.0000	0.5488	0.7857	8.0000
5.0000	8.0000	8.0000	8.0000	0	0.5488	0.7857	8.0000
5.0000	9.0000	8.0000	8.0000	0	0.5488	0.7857	8.0000
5.0000	10.0000	8.0000	8.0000	0	0.5488	0.7857	8.0000
4.1823	1.0000	8.0000	8.0000	0	0.5488	0.7857	8.0000
4.1823	2.0000	8.0000	8.0000	0	0.5488	0.7857	8.0000
4.1823	3.0000	9.0000	8.0000	1.0000	0.7873	0.7590	9.0000
4.1823	4.0000	6.0000	9.0000	-3.0000	0.7873	0.7590	6.0000
4.1823	5.0000	6.0000	6.0000	0	0.7873	0.7590	6.0000
4.1823	6.0000	6.0000	6.0000	0	0.7873	0.7590	6.0000
4.1823	7.0000	8.0000	6.0000	2.0000	0.6199	0.8609	6.0000
4.1823	8.0000	8.0000	8.0000	0	0.6199	0.8609	8.0000
4.1823	9.0000	6.0000	8.0000	-2.0000	0.6199	0.8609	6.0000
4.1823	10.0000	9.0000	6.0000	3.0000	0.4881	0.0678	9.0000
2.9490	1.0000	9.0000	9.0000	0	0.4881	0.0678	9.0000
2.9490	2.0000	8.0000	9.0000	-1.0000	0.4881	0.0678	8.0000
2.9490	3.0000	6.0000	8.0000	-2.0000	0.4881	0.0678	6.0000
2.9490	4.0000	8.0000	6.0000	2.0000	0.5075	0.3490	8.0000
2.9490	5.0000	8.0000	8.0000	0	0.5075	0.3490	8.0000
2.9490	6.0000	6.0000	8.0000	-2.0000	0.5075	0.3490	6.0000
2.9490	7.0000	6.0000	6.0000	0	0.5075	0.3490	6.0000
2.9490	8.0000	6.0000	6.0000	0	0.5075	0.3490	6.0000
2.9490	9.0000	8.0000	6.0000	2.0000	0.5075	0.4106	8.0000
2.9490	10.0000	9.0000	8.0000	1.0000	0.7124	0.9525	8.0000
0.1516	1.0000	9.0000	9.0000	0	0.7124	0.9525	9.0000
0.1516	2.0000	8.0000	9.0000	-1.0000	0.7124	0.9525	8.0000
0.1516	3.0000	6.0000	8.0000	-2.0000	0.7124	0.9525	6.0000
0.1516	4.0000	6.0000	6.0000	0	0.7124	0.9525	6.0000
0.1516	5.0000	6.0000	6.0000	0	0.7124	0.9525	6.0000
0.1516	6.0000	8.0000	6.0000	2.0000	0.0000	0.4231	6.0000
0.1516	7.0000	8.0000	8.0000	0	0.0000	0.4231	8.0000
0.1516	8.0000	8.0000	8.0000	0	0.0000	0.4231	8.0000
0.1516	9.0000	8.0000	8.0000	0	0.0000	0.4231	8.0000
0.1516	10.0000	8.0000	8.0000	0	0.0000	0.4231	8.0000
0.0357	1.0000	9.0000	8.0000	1.0000	0.0000	0.5081	8.0000
0.0357	2.0000	9.0000	9.0000	0	0.0000	0.5081	9.0000
0.0357	3.0000	8.0000	9.0000	-1.0000	0.0000	0.5081	8.0000
0.0357	4.0000	9.0000	8.0000	1.0000	0.0000	0.4281	8.0000
0.0357	5.0000	9.0000	9.0000	0	0.0000	0.4281	9.0000
0.0357	6.0000	8.0000	9.0000	-1.0000	0.0000	0.4281	8.0000
0.0357	7.0000	6.0000	8.0000	-2.0000	0.0000	0.4281	6.0000
0.0357	8.0000	8.0000	6.0000	2.0000	0.0000	0.8947	6.0000
0.0357	9.0000	6.0000	8.0000	-2.0000	0.0000	0.8947	6.0000
0.0357	10.0000	9.0000	6.0000	3.0000	0.0000	0.7763	6.0000
0.0271	1.0000	8.0000	9.0000	-1.0000	0.0000	0.7763	8.0000
0.0271	2.0000	9.0000	8.0000	1.0000	0.0000	0.6897	8.0000
0.0271	3.0000	8.0000	9.0000	-1.0000	0.0000	0.6897	8.0000
0.0271	4.0000	8.0000	8.0000	0	0.0000	0.6897	8.0000
0.0271	5.0000	6.0000	8.0000	-2.0000	0.0000	0.6897	6.0000
0.0271	6.0000	8.0000	6.0000	2.0000	0.0000	0.5262	6.0000
0.0271	7.0000	8.0000	8.0000	0	0.0000	0.5262	8.0000
0.0271	10.0000	9.0000	6.0000	3.0000	0.0000	0.6434	6.0000

Tempe.	Loop	current sol	old_sol	deferens	prop	rand	best_sol
0.0112	1.0000	9.0000	9.0000	0	0.0000	0.6434	9.0000
0.0112	2.0000	8.0000	9.0000	-1.0000	0.0000	0.6434	8.0000
0.0112	3.0000	9.0000	8.0000	1.0000	0.0000	0.4899	8.0000
0.0112	4.0000	9.0000	9.0000	0	0.0000	0.4899	9.0000
0.0112	5.0000	9.0000	9.0000	0	0.0000	0.4899	9.0000
0.0112	6.0000	6.0000	9.0000	-3.0000	0.0000	0.4899	6.0000
0.0112	7.0000	8.0000	6.0000	2.0000	0.0000	0.8155	6.0000
0.0112	8.0000	6.0000	8.0000	-2.0000	0.0000	0.8155	6.0000
0.0112	9.0000	9.0000	6.0000	3.0000	0.0000	0.7899	6.0000
0.0112	10.0000	6.0000	9.0000	-3.0000	0.0000	0.7899	6.0000
0.0093	1.0000	8.0000	6.0000	2.0000	0.0000	0.2660	6.0000
0.0093	2.0000	9.0000	8.0000	1.0000	0.0000	0.6400	6.0000
0.0093	3.0000	8.0000	9.0000	-1.0000	0.0000	0.6400	8.0000
0.0093	4.0000	8.0000	8.0000	0	0.0000	0.6400	8.0000
0.0093	5.0000	8.0000	8.0000	0	0.0000	0.6400	8.0000
0.0093	6.0000	6.0000	8.0000	-2.0000	0.0000	0.6400	6.0000
0.0093	7.0000	6.0000	6.0000	0	0.0000	0.6400	6.0000
0.0093	8.0000	6.0000	6.0000	0	0.0000	0.6400	6.0000
0.0093	9.0000	6.0000	6.0000	0	0.0000	0.6400	6.0000
0.0093	10.0000	8.0000	6.0000	2.0000	0.0000	0.8896	6.0000
0.0020	1.0000	9.0000	8.0000	1.0000	0.0000	0.4642	6.0000
0.0020	2.0000	8.0000	9.0000	-1.0000	0.0000	0.4642	8.0000
0.0020	3.0000	8.0000	8.0000	0	0.0000	0.4642	8.0000
0.0020	4.0000	6.0000	8.0000	-2.0000	0.0000	0.4642	6.0000
0.0020	5.0000	9.0000	6.0000	3.0000	0	0.2737	6.0000
0.0020	6.0000	8.0000	9.0000	-1.0000	0	0.2737	8.0000
0.0020	7.0000	9.0000	8.0000	1.0000	0.0000	0.7346	8.0000
0.0020	8.0000	6.0000	9.0000	-3.0000	0.0000	0.7346	6.0000
0.0020	9.0000	8.0000	6.0000	2.0000	0	0.7054	6.0000
0.0020	10.0000	6.0000	8.0000	-2.0000	0	0.7054	6.0000
0.0009	1.0000	9.0000	6.0000	3.0000	0	0.6825	6.0000
0.0009	2.0000	8.0000	9.0000	-1.0000	0	0.6825	8.0000
0.0009	3.0000	9.0000	8.0000	1.0000	0	0.7017	8.0000
0.0009	4.0000	8.0000	9.0000	-1.0000	0	0.7017	8.0000
0.0009	5.0000	8.0000	8.0000	0	0	0.7017	8.0000
0.0009	6.0000	8.0000	8.0000	0	0	0.7017	8.0000
0.0009	7.0000	8.0000	8.0000	0	0	0.7017	8.0000
0.0009	8.0000	9.0000	8.0000	1.0000	0	0.3459	8.0000
0.0009	9.0000	9.0000	9.0000	0	0	0.3459	9.0000
0.0009	10.0000	8.0000	9.0000	-1.0000	0	0.3459	8.0000
0.0004	1.0000	8.0000	8.0000	0	0	0.3459	8.0000
0.0004	2.0000	8.0000	8.0000	0	0	0.3459	8.0000
0.0004	3.0000	6.0000	8.0000	-2.0000	0	0.3459	6.0000
0.0004	4.0000	8.0000	6.0000	2.0000	0	0.4826	6.0000
0.0004	5.0000	8.0000	8.0000	0	0	0.4826	8.0000
0.0004	6.0000	9.0000	8.0000	1.0000	0	0.6987	8.0000
0.0004	7.0000	8.0000	9.0000	-1.0000	0	0.6987	8.0000
0.0004	8.0000	8.0000	8.0000	0	0	0.6987	8.0000
0.0004	9.0000	9.0000	8.0000	1.0000	0	0.7090	8.0000
0.0004	10.0000	8.0000	9.0000	-1.0000	0	0.7090	8.0000
0.0001	1.0000	8.0000	8.0000	0	0	0.7090	8.0000
0.0001	2.0000	6.0000	8.0000	-2.0000	0	0.7090	6.0000
0.0001	3.0000	8.0000	6.0000	2.0000	0	0.0799	6.0000
0.0001	4.0000	9.0000	8.0000	1.0000	0	0.9067	6.0000
0.0001	5.0000	9.0000	9.0000	0	0	0.9067	9.0000
0.0001	6.0000	9.0000	9.0000	0	0	0.9067	9.0000
0.0001	7.0000	8.0000	9.0000	-1.0000	0	0.9067	8.0000
0.0001	8.0000	9.0000	8.0000	1.0000	0	0.5343	8.0000
0.0001	9.0000	6.0000	9.0000	-3.0000	0	0.5343	6.0000
0.0001	10.0000	8.0000	6.0000	2.0000	0	0.8672	6.0000