Using Mathematical Optimization Method (Tabu Search) for Edges Crossing Points Reduction in Graphs

استخدام طريقة الامثلية الرياضية (البحث المحرم) لتقليل نقاط تقاطع الحواف في المخططات

Assistant Lecture/ Enaam H. Abd
Kerbalaa University / College of Science
Computer Department

Abstract

A graph is an algebraic structure of nodes and edges is commonly presented geometrically by diagrams depicting nodes as dots and edges as curves or straight line segments that connect the dots . Some applications need to reduce the number of edges crossing such as Telecommunications e.g. spanning trees ,Vehicle routing e.g. Roads without underpasses, VLSI e.g. computer chips , and Road networks. This aim of this paper is minimize the number of crossing points in graphs by applying the mathematical optimization (Tabu Search) method. More important results was recorded by implement this method .

المستخلص

المخطط هو هيكل رياضي يتكون من عدد من العقد والحواف تمثل هندسيا بواسطة نقاط بالنسبة للعقد وحواف التي تمثل بواسطة المنحنيات او الخطوط المستقيمة التي تربط العقد . بعض التطبيقات تحتاج الى تقليل عدد التقاطعات بين الحواف مثلا في الاتصالات ، الدوائر لالكترونية ، وشبكات الطرق ...الخ .

يهدف البحث الى تقليل عدد نقاط التقاطع في المخططات بواسطة تطبيق احد طرائق الامثلية

(وهي طريقة Tabu Search) حيث تم الحصول على نتائج مهمة من تطبيق هذه الطريقة

1. Principle of Optimization Problems (in mathematics)

In mathematics, the term optimization, or mathematical programming, refers to the study of problems in which one seeks to minimize or maximize a real function by systematically choosing the values of real or integer variables from within an allowed set. This problem can be represented in the following way :

Given: a function $f : A \longrightarrow \mathbb{R}$ from some set A to the real numbers

Sought: an element x_0 in A such that $f(x_0) \le f(x)$ for all x in A ("minimization") or such that $f(x_0) \ge f(x)$ for all x in A ("maximization").

Such a formulation is called an optimization problem or a mathematical programming problem (a term not directly related to computer programming, but still in use for example in linear programming). Many real-world and theoretical problems may be modeled in this general framework. Problems formulated using this technique in the fields of physics and computer vision may refer to the technique as energy minimization, speaking of the value of the function *f* as representing the energy of the system being modeled

Typically, *A* is some subset of the Euclidean space R^n, often specified by a set of *constraints*, equalities or inequalities that the members of *A* have to satisfy. The elements of *A* are called *feasible solutions*. The function *f* is called an *objective function*, or *cost function*. A feasible solution that minimizes (or maximizes, if that is the goal) the objective function is called an *optimal solution*. [9]
The domain *A* of *f* is called the *search space*, while the elements of *A* are called *candidate solutions* or *feasible solutions*.
Generally, when the feasible region or the objective function of the problem does not present convexity, there may be several local minima and maxima, where a *local minimum* x^* is defined as a point for which there exists some $\delta > 0$ so that for all x such that

$$\|\mathbf{x}-\mathbf{x}^*\| \leq \delta;$$

the expression

$$f(\mathbf{x}^*) \leq f(\mathbf{x})$$

holds; that is to say, on some region around x^* all of the function values are greater than or equal to the value at that point. Local maxima are defined similarly.

A large number of algorithms proposed for solving non-convex problems – including the majority of commercially available solvers – are not capable of making a distinction between local optimal solutions and rigorous optimal solutions, and will treat the former as actual solutions to the original problem. The branch of applied mathematics and numerical analysis that is concerned with the development of deterministic algorithms that are capable of guaranteeing convergence in finite time to the actual optimal solution of a non-convex problem is called global optimization .
Here are a popular techniques (methods) for dealing with optimization problems :[9]

- Hill climbing
- Simulated annealing
- Quantum annealing
- Tabu search
- Beam search
- Genetic algorithms
- Ant colony optimization
- Evolution strategy
- Stochastic tunneling
- Differential evolution
- Particle swarm optimization
- Harmony search

Our research about the Tabu Search method, which is a mathematical optimization method, belonging to the class of local search techniques.

Some details of Tabu Search method was explained in section 5 , the following section show the introduction to Graph Theory and the method of computing edges crossing points in graph .

2. Introduction to Graph theory

A graph which is essentially an algebraic structure of nodes and edges is commonly presented geometrically by diagrams depicting nodes as dots and edges as curves or straight line segments that connect the dots [6] .The earliest use of graph theoretical methods probably goes back to the 18th century. At this time, there were seven bridges crossing the river Pregel in the town of Königsberg. The folks had long amused themselves with the following problem: Is it possible to walk through the town using every bridge just once, and returning home at the end? The problem was solved by Leonhardt Euler (1707–1783) in 1736 by mapping it to a graph problem and solving it for arbitrary graphs [2] .

A graph is a collection of vertices or nodes, pairs of which are joined by lines or edges. A graph G= (V, E) is an ordered pair of finite sets V and E. The elements of V are called vertices (vertices are also called nodes and points). The elements of E are called edges (edges are also called arcs and lines). Each edge in E joins two different vertices of V and is denoted by (i, j), where i and j are the two vertices joined by E . Graphs can be used not only to represent physical relationships ,but also to represent logical relationships , biological relationships ,arithmetic relationships [5] .

We represent a graph or digraphs by adjacency matrix of an n-vertex graph G= (V, E) is an $n \times n$ matrix A. Each element of A is either zero or one . We shall assume that V= (1, 2,… n) .

Some applications need to reduce the number of edges crossing such as Telecommunications e.g. spanning trees ,Vehicle routing e.g. Roads without underpasses, VLSI e.g. computer chips , and Road networks. .The following section show the details of computes the number of edges crossing.

3. Compute Edges Crossing Points in a Graph

Many approaches of computing graph crossing points founded focus on directed graph , after initialization of graph ,level pairs can be sorted in descending order according to a number of connections between the level pairs .Evaluation of the graph can progress according to the order of the level pairs so that those pairs likely to have the greatest number of connections are processed first .[8]

Another approach is the Sugiyama algorithm initially changes a graph into a “proper” hierarchical graph that has no cycles, and only has edges between adjacent levels. That is, cycles are removed from the graph by reversing and marking edges that cause cycles and inserting dummy nodes to ensure that no edges span more than one node. Each pair of levels in the layered graph is taken sequentially and a connection matrix is calculated, with which a number of crossings can be computed. A count for each pair of levels is accumulated and a total of crossing counts for each level pair is taken as the total number of edge crossings for the graph.[8]

In this paper , the following approach can be applied in directed and undirected graph depending on algorithm for determining the intersection point of two edges (or line segments) in 2 dimensions using the following equations , show Figure (1) [4].

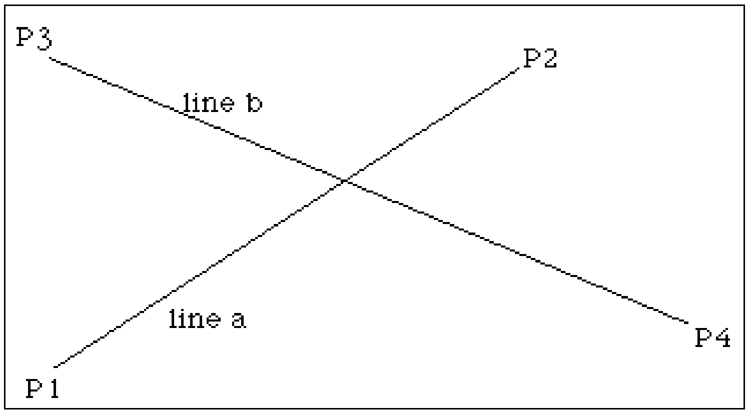


Figure (1) two line segments intersected

The equations of the lines are [4] :

$$Pa = P1 + u_a (P2 - P1) \quad \dots (1)$$

$$Pb = P3 + u_b (P4 - P3) \quad \dots (2)$$

Solving for the point where $P_a = P_b$ gives the following two equations in two unknowns (u_a and u_b)

$$x1 + u_a (x2 - x1) = x3 + u_b (x4 - x3) \quad \dots (3)$$

and

$$y1 + u_a (y2 - y1) = y3 + u_b (y4 - y3) \quad \dots (4)$$

Solving gives the following expressions for u_a and u_b

$$u_a = \frac{(x4 - x3)(y1 - y3) - (y4 - y3)(x1 - x3)}{(y4 - y3)(x2 - x1) - (x4 - x3)(y2 - y1)} \quad \dots (5)$$

$$u_b = \frac{(x2 - x1)(y1 - y3) - (y2 - y1)(x1 - x3)}{(y4 - y3)(x2 - x1) - (x4 - x3)(y2 - y1)}$$

Substituting either of these into the corresponding equation for the line gives the intersection point. For example the intersection point (x,y) is

$$x = x1 + u_a (x2 - x1) \quad \dots (6)$$

$$y = y1 + u_a (y2 - y1) \quad \dots (7)$$

The denominators for the equations for ua and ub are the same. If the denominator for the equations for u_a and u_b is 0 then the two lines are parallel. If the denominator and numerator for the equations for u_a and u_b are 0 then the two lines are coincident. The equations apply to lines, if the intersection of line segments is required then it is only necessary to test if u_a and u_b lie between 0 and 1. Whichever one lies within that range then the corresponding line segment contains the intersection point. If both lie within the
range of 0 to 1 then the intersection point is within both line segments[4].
These steps repeated on all lines of the graph .

4. Representation of Graph

In this section we try to describe the representation of graph in such a way that facilitate the dealing with it and support reliability in implementation, processing, and outcoming results. The graph is generated randomly by the matrix of connection (1 indicate connection and 0 indicate no connection between two nodes) and represented as a vector which consists of index(node number), as well as the x and y coordinates for all graph nodes on the screen. The coordinates will be generated randomly. See figure (2)

Index	1	2	3	4	5	6	7	8	9	10
X coordinates	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
Y coordinates	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	Y10

Fig. (2) Graph represented as a vector

Index field important in processing and the coordinate field used in visualization the graph.

5. Description of Tabu Search Method

Tabu search is a mathematical optimization method, belonging to the class of neighborhood search techniques. Tabu search enhances the performance of neighborhood search by using memory structures [7].

The word “tabu” comes from a Polynesian language where it indicates things that cannot be touched because they are sacred. A blind neighborhood search is doomed to entrapment in local optima. Therefore, it may be possible to enhance this search by avoiding certain moves that lead to such entrapment. These moves are called “tabu”, hence the name of the method [1] .

The idea of tabu search is simple. It starts with a random initial solution S, like simple neighborhood search does. At each step, the best solution S’ from the neighborhood of S is chosen (note that best solution mean graph or network with minimum edges crossing number), regardless of the relationship between S and S’. That is, if S’ is worse than S, it is still chosen, if it is the best solution in the neighborhood of S. After choosing S’ as the next current solution, the modification brought to S in order to obtain S’ is recorded in a memory called the “tabu list”. The tabu list contains the K most recent S→S’ moves, where K is the list dimension. Then, to rephrase a previous statement with respect to newly introduced terms, at each step of the algorithm, the best solution S’ from the neighborhood of S is chosen which also satisfies the condition that the move from S to S’ is not a tabu move. After choosing the next solution S’ the tabu list is altered, such that the oldest entry is dismissed and the most recent S→S’ move taken is inserted (the tabu list may be implemented, therefore, as a queue). Tabu conditions may sometimes be too drastic. They may forbid moves that lead to unvisited and potentially attractive solutions. For example, if a tabu move would produce a solution which is better than the best found so far, the tabu status of that move is dropped [1].
Consider the example of a graph with N = 10 where N is the number of nodes, see figure(3)

Index	1	2	3	4	5	6	7	8	9	10
X coordinates	200	200	50	250	300	270	180	200	100	70
Y coordinates	50	400	300	200	300	100	200	300	100	200

Fig. (3) Example of graph representation

We show that node1 in coordinate x1 =200 , y1=50 ; node2 in coordinate x2=200,y2=400, and so on …. Since each permutations of nodes [1-10] are graphs, that mean the number of possible graphs is (10!).

If we have the following graph vector :

Index	2	1	3	4	5	6	7	8	9	10
X coordinates	200	200	50	250	300	270	180	200	100	70
Y coordinates	50	400	300	200	300	100	200	300	100	200

Fig. (4) Example of graph representation when applying Tabu search method

For more explanation see figure(5) after one iteration of algorithm

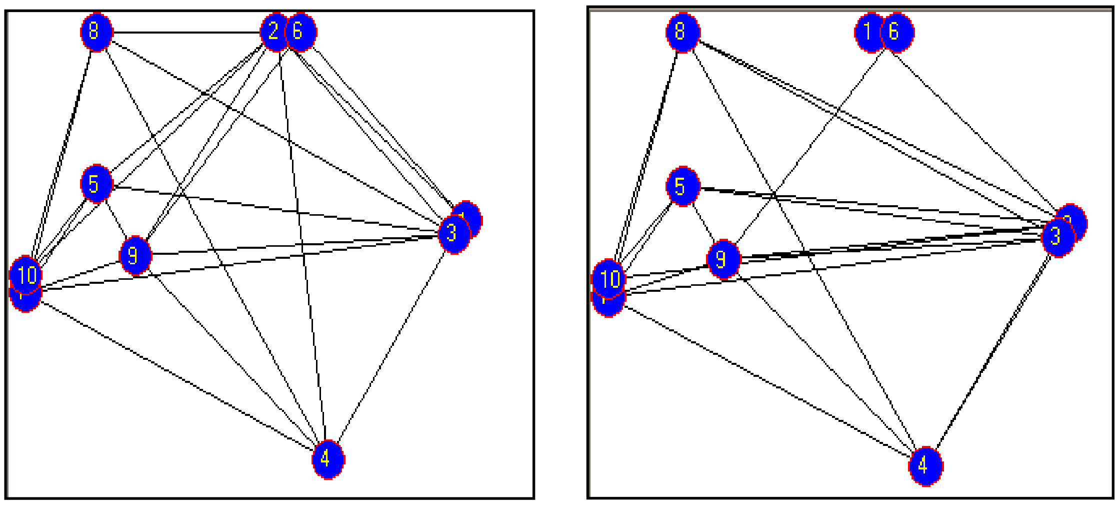


Fig. (5) Visualization of Graph after one iteration of Tabu search method

That mean the node1 replaced by node2 , node2 replaced by node 1 , and so on … .

For each (i, j) pair with i < j, a different neighbor is obtained. The tabu search algorithm states that the best of all the neighbors of S which does not imply tabu moves is chosen for the next iteration. Then, the oldest tabu move is dismissed from the tabu list and the most recent move is inserted as a tabu move. For this algorithm, a tabu move could be considered the pair (i, j) that led to obtaining the best S’ from S. The idea is that for the next K iterations (where K is the tabu tenure or the length of the tabu list, another move dictated by the same pair (i, j) is forbidden).[1]

The tabu tenure should be carefully chosen. A very big tabu tenure will end up in forbidding all moves, because all possible (i, j) pairs will eventually be in the tabu list. A very small tabu tenure will result in the tabu list being of little or no use, because it is very unlikely that the same (i, j) pair is randomly chosen within the next very few iterations. If each (i, j) pair leads to a different neighbor (where $i < j \leq N$) then the number of possible neighbors is given by $R = N \cdot (N - 1) / 2$.
The termination condition may involve completion of a certain number of iterations .

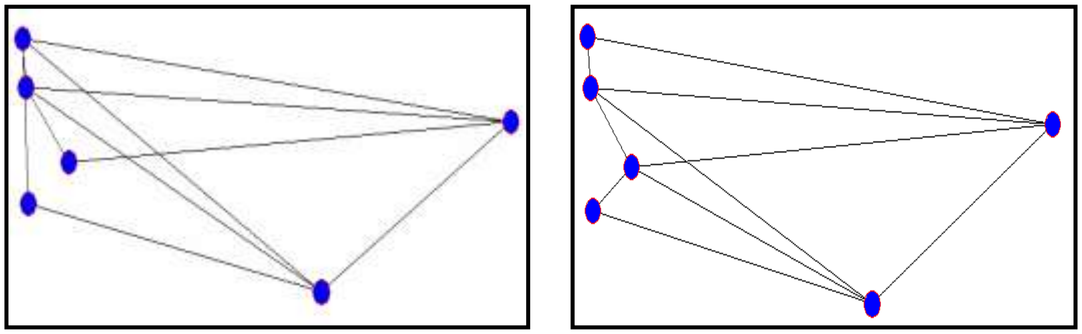
The summarized of this method explained in the following algorithm[3] :

```
Tabu Search Algorithm
Inputs:  Graph Matrix
Output: Best Solution
Begin Of Algorithm
 Initial solution
While termination criterion not reached
 Begin
       Neighborhood generation
       Evaluation of neighborhood
       Selection of best solution
       Update tabu list
       New solution
 End
End Of Algorithm
```

6. Implementation and Experimental Results

In this paper, applying the Tabu search method with various number of graph nodes , the matrix of connection and the coordinates of each node will be generated randomly .The results yields with number of nodes (6,10,15, and 25) .two experiment , first with 10 iteration ,and the second with 30 iteration .

First Experiment (10 iteration) :



a. Before (No. of Crossing =3) b. After (No. of Crossing =1)

Fig. (6) Visualization of Graph (with 6 nodes) and (10 iterations)

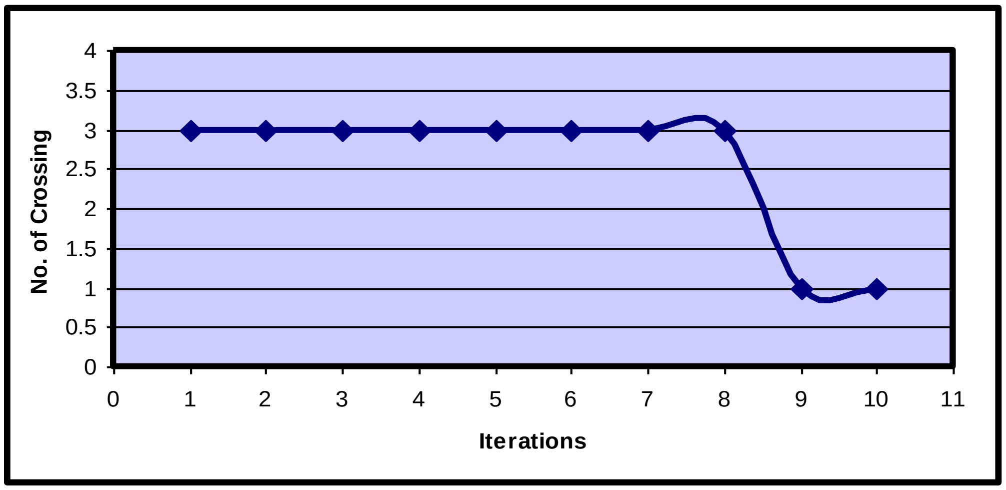
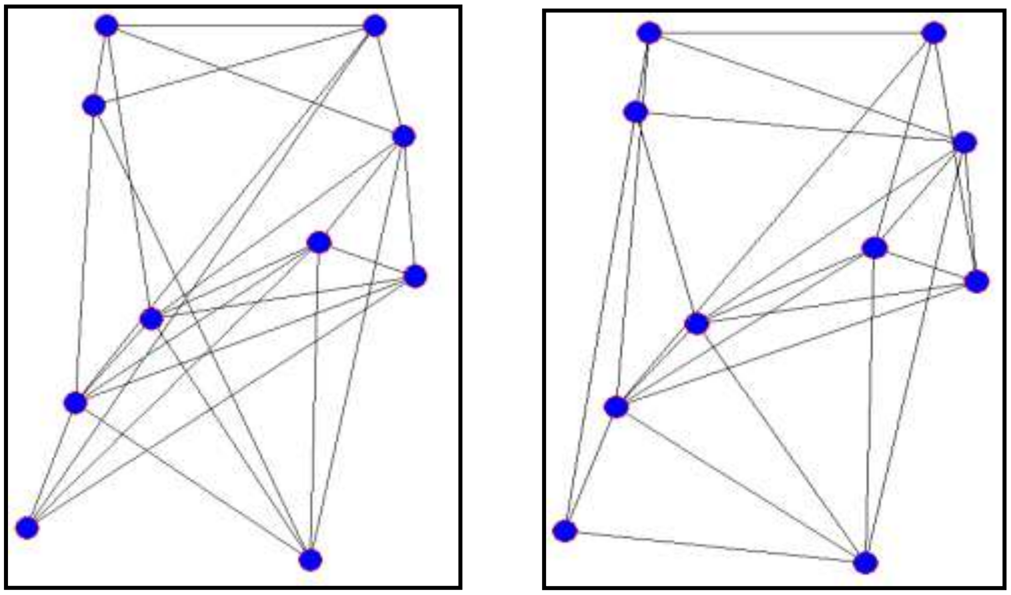


Fig. (7) Result of algorithm (with 6 nodes) and (10 iterations)



a. Before (No. of Crossing =38) b. After (No. of Crossing =23)

Fig. (8) Visualization of Graph (with 10 nodes) and (10 iterations)

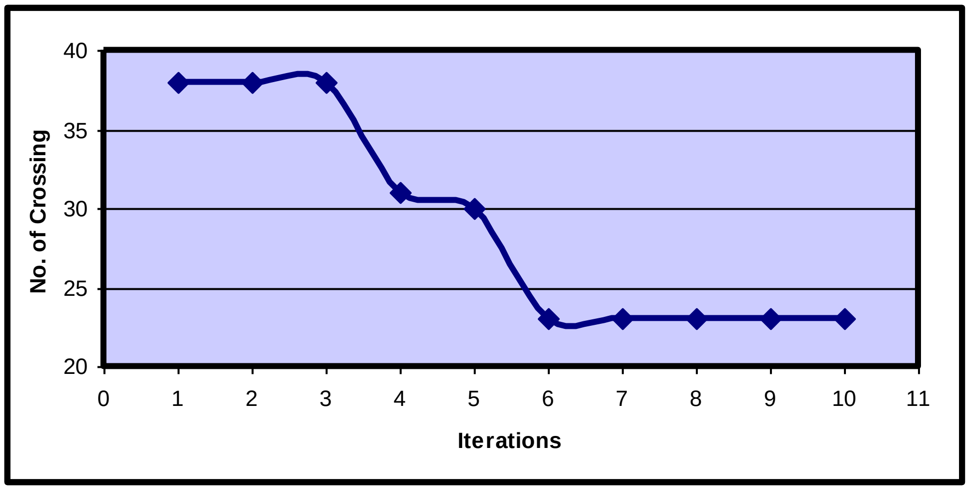
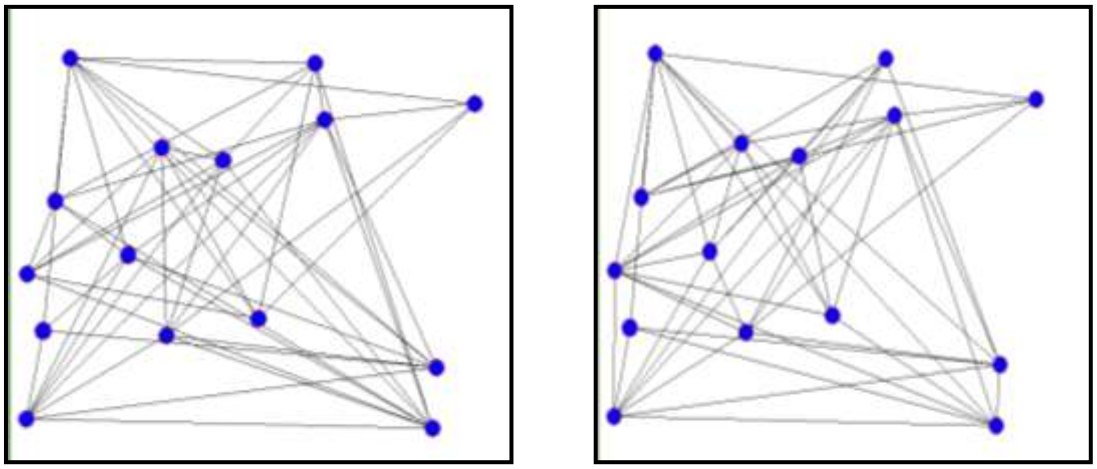


Fig. (9) Result of algorithm (with 10 nodes) and (10 iterations)



a. Before (No. of Crossing =214) b. After (No. of Crossing =168)

Fig. (10) Visualization of Graph (with 15 nodes) and (10 iterations)

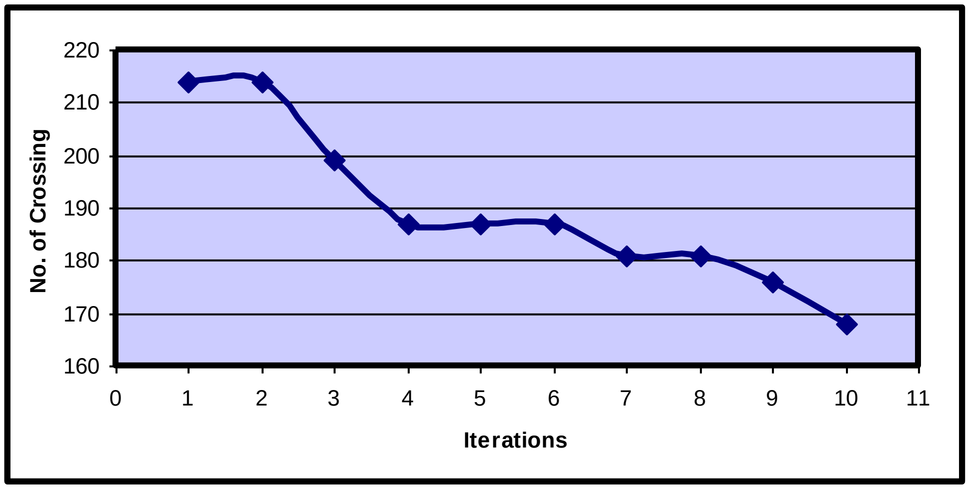


Fig. (11) Result of algorithm (with 15 nodes) and (10 iterations)

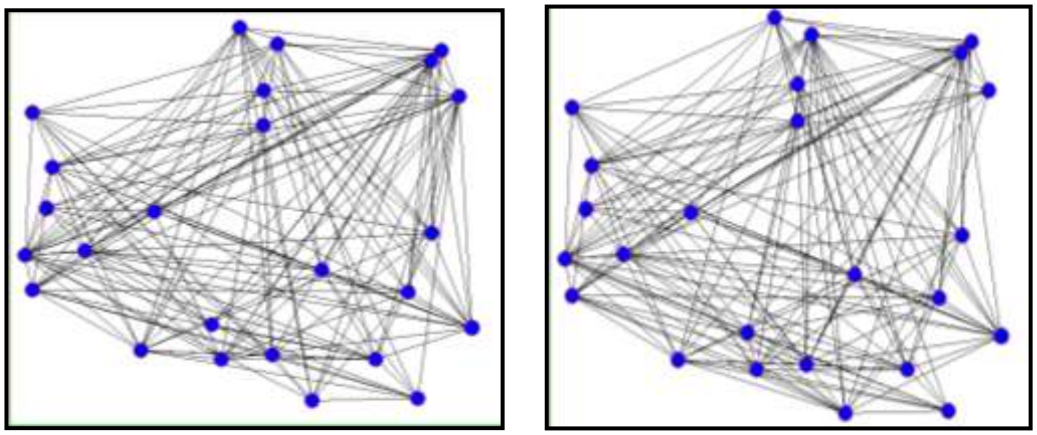

a. Before (No. of Crossing =2426) b. After (No. of Crossing =2243)

Fig. (12) Visualization of Graph (with 25 nodes) and (10 iterations)

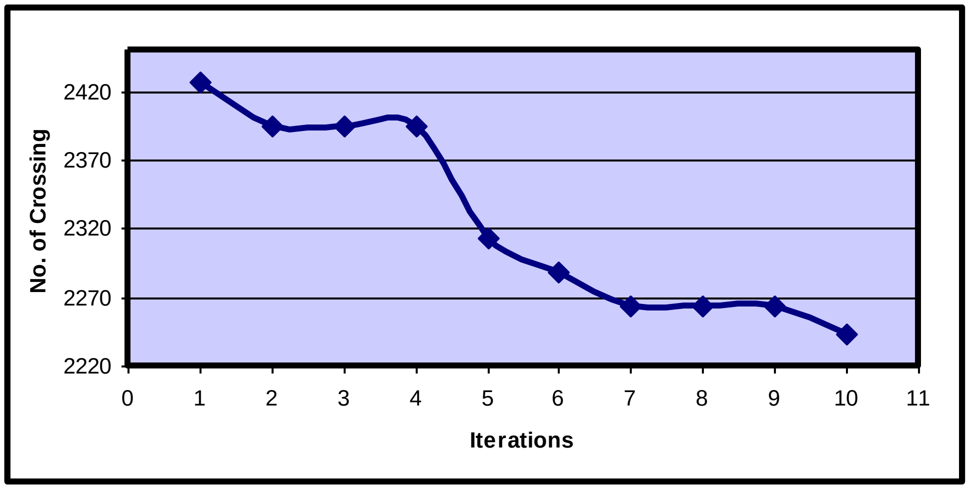


Fig. (13) Result of algorithm (with 25 nodes) and (10 iterations)

Second Experiment (30 iteration) :

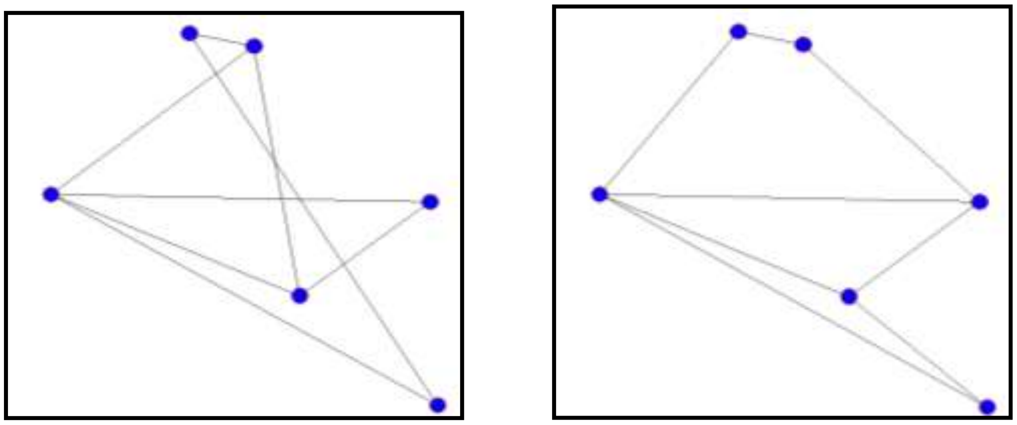

a. Before (No. of Crossing =5) b. After (No. of Crossing =0)

Fig. (14) Visualization of Graph (with 6 nodes) and (30 iterations)

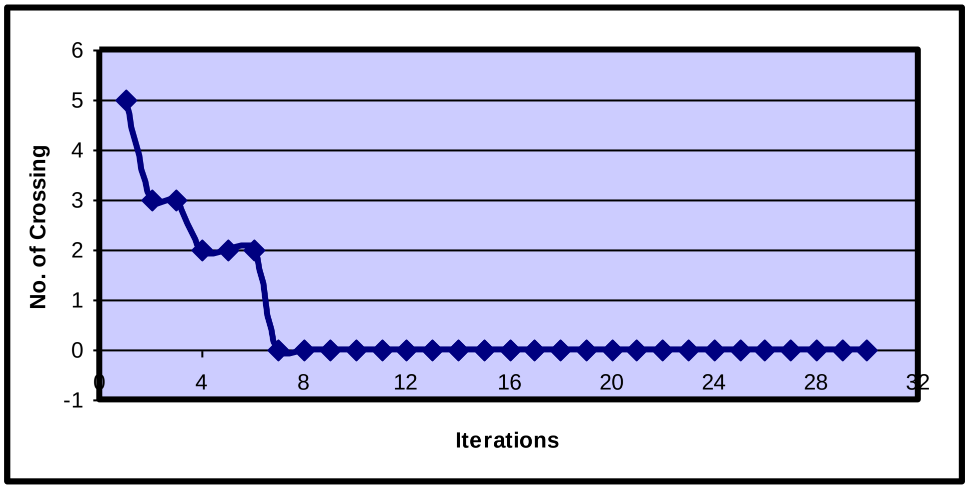
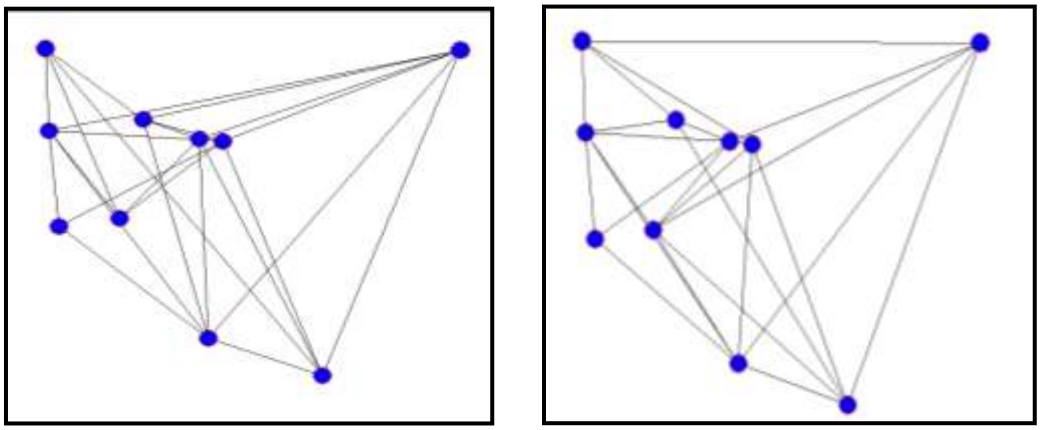


Fig. (15) Result of algorithm (with 6 nodes) and (30 iterations)



a. Before (No. of Crossing =29) b. After (No. of Crossing =15)

Fig. (16) Visualization of Graph (with 10 nodes) and (30 iterations)

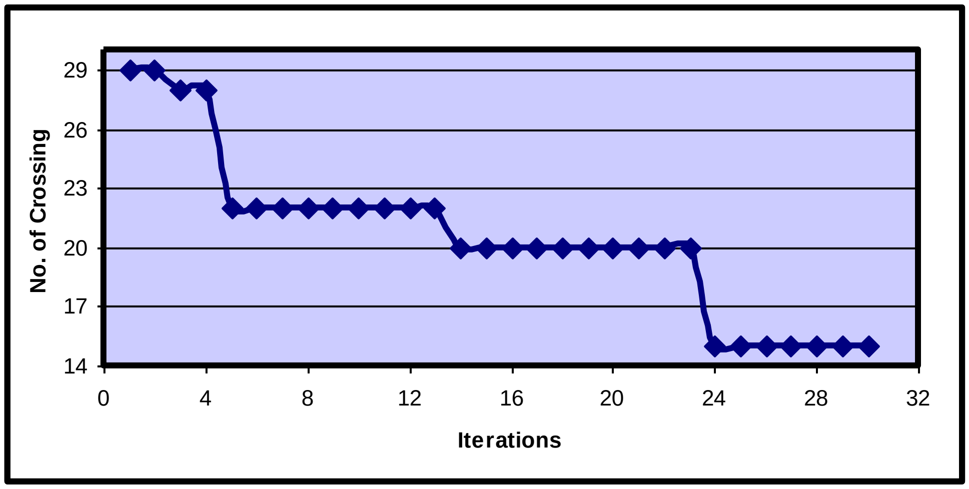
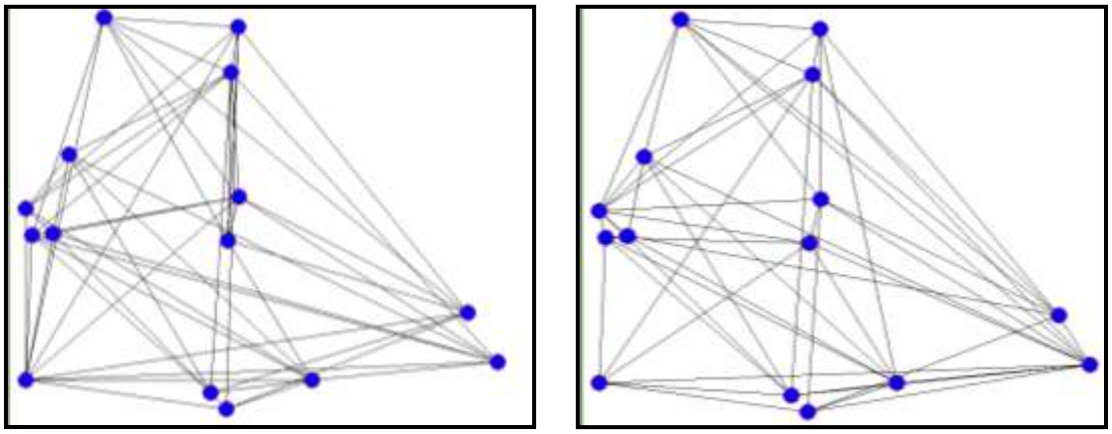


Fig. (17) Result of algorithm (with 10 nodes) and (30 iterations)



a. Before (No. of Crossing =208) b. After (No. of Crossing =126)

Fig. (18) Visualization of Graph (with 15 nodes) and (30 iterations)

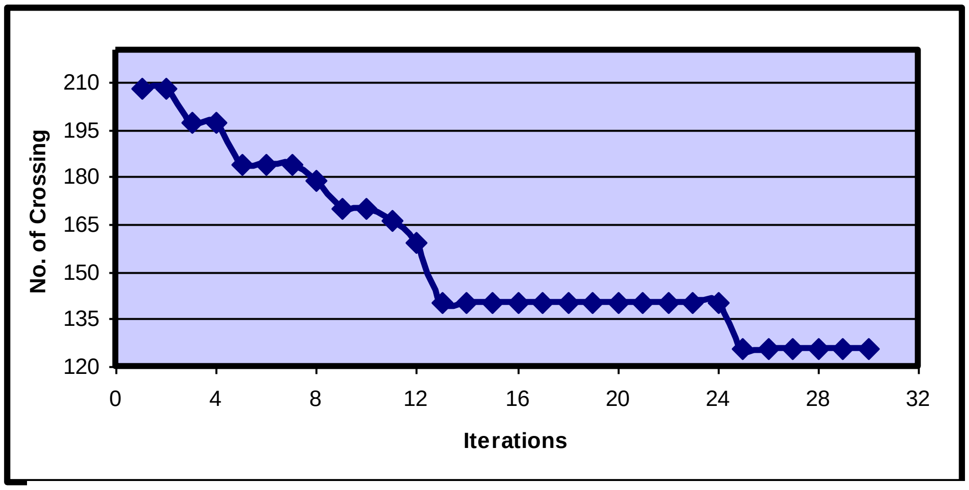
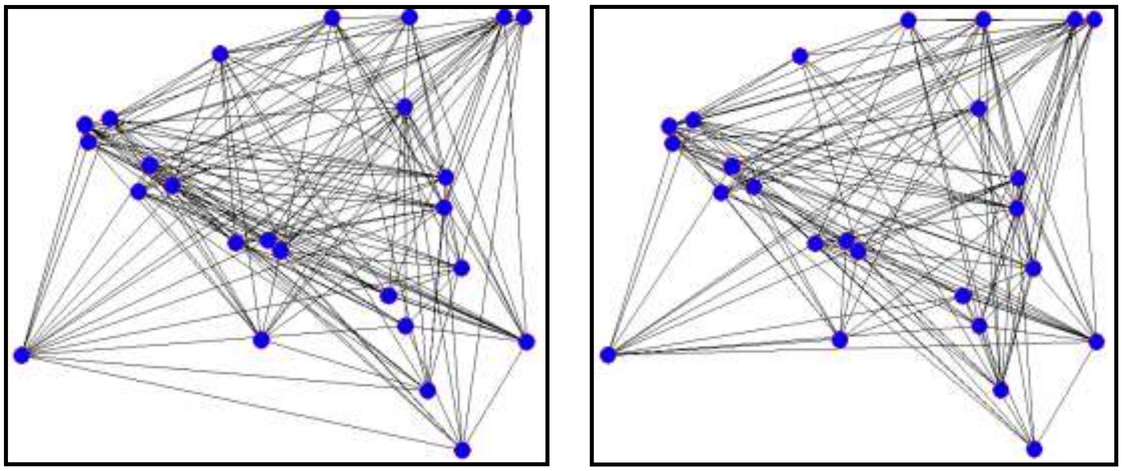


Fig. (19) Result of algorithm (with 15 nodes) and (30 iterations)



a. Before (No. of Crossing =2559) b. After (No. of Crossing =1898)

Fig. (20) Visualization of Graph (with 25 nodes) and (30 iterations)

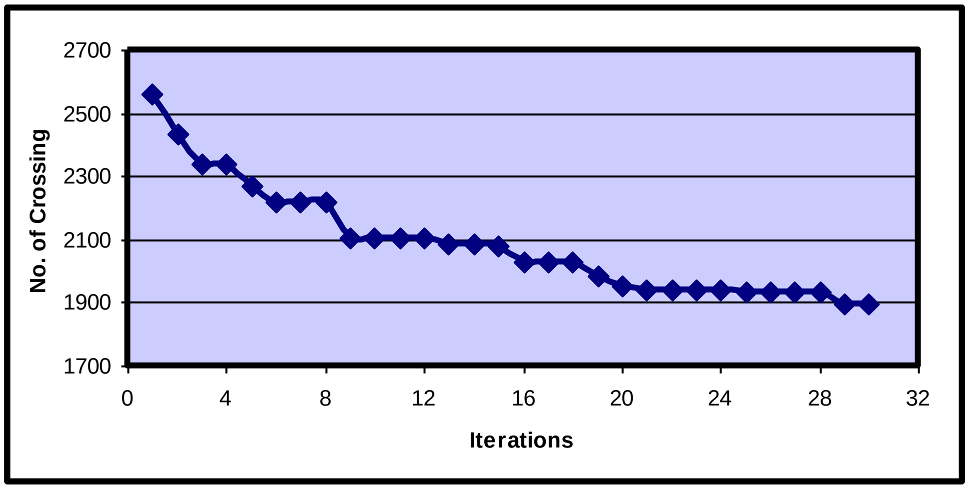


Fig. (21) Result of algorithm (with 25 nodes) and (30 iterations)

The ratio of rest crossing point in a Graph can be listed in table (1)

Table(1) The ratio of rest crossing point in a Graph

Nodes Number	Ratio of Rest Crossing Points	
	10 Iterations	30 Iterations
6	0.3	0
10	0.6	0.5
15	0.78	0.6
25	0.92	0.74
Total	0.65	0.46

7. Conclusions

In an optimization problem, the types of mathematical relationships between the objective and constraints and the decision variables determine how hard it is to solve, the solution methods or algorithms that can be used for optimization, and the confidence you can have that the solution is truly optimal.

From previous description of Tabu Search algorithm and implementation of it , The Tabu search can explore multi solutions sometimes lead to optimal solution .Good results can obtained for more iteration as we show from table(1) , the problem as simply is reordering the locations of nodes to reduce the number of edges crossing points (as shown from previous figures), this objective was implemented by effective method (Tabu Serch) which enable to replacement between nodes .More complexity occur when increase the number of nodes distributed in small graph area .

References

[1] A . Hertz1, D.Werra1 ,A TUTORIAL ON TABU SEARCH,2004

[2] Alexander K. Hartmann and Martin Weigt, Phase Transitions in Combinatorial Optimization Problems ,2008.

[3] N.Evin , Hurestic Search ,2003 .

[4] Paul Bourke , Intersection point of two lines (2 dimensions) ,1998 .

[5] S.Base, Computer Algorithm , Dison western, publishing company ,1988.

[6] www.cut-the-knot.org/do_you_know/CrossingNumber.shtml

[7] www.en-wikipedia.org

[8] www.freepatentsonline.com/y2010/0073375.html

[9]www.statemaster.com/encyclopedia/Optimization