Proposal of Backtracked Tabu Search (BTS) Algorithm

Dr. Ahmed Tariq Sadiq *

Received on: 2/2/2009
Accepted on: 5/11/2009

Abstract

There are several heuristic search techniques, Tabu Search (TS) is one of them. TS based on generating the neighbor states but it has several problems in its work. This paper presents a new version of TS based on backtracking techniques (called BTS) to reduce the effect of these problems. Two case studies have been solved using BTS, 4-Color Map and Traveling Salesman Problem. The proposed algorithm gives good results compare with the original, the iteration numbers are less and the local minimum or non-optimal solutions are less.

أقتراح خوارزمية البحث المحرم المتراجعة

الخلاصة

هناك عدة تقنيات طرق بحث ارشادية، البحث المحرم يمثل احدها. البحث المحرم يستند على توليد جيل من أفضل الجيران ولكنه يحوي عدة مشاكل في عمله. البحث المقدم يمثل نسخة جديدة من البحث المحرم المستند على تقنية التراجع لتقليل تأثير تلك المشاكل. تم حل مشكلتين كدراسة حالة هما مشكلة خريطة الالوان الاربعة ومشكلة البائع المتجول. الخوارزمية المقترحة أعطت نتائج أفضل بالمقارنة مع الاصلية فعدد دورات التنفيذ كان أقل وكذلك تم تقليل حالات الحل الذي هو ليس بالافضل ومشكلة الامثلية الصغرى المحلية.

1- Introduction

A huge collection of optimization techniques have been suggested by a crowd of researchers of different fields; an infinity of refinements have made these techniques work on specific types of applications. All these procedures based on some common ideas and are furthermore characterized by a few additional specific features. Among the optimization procedures, the iterative techniques play an important role; for most optimization problems no procedure is known in general to get directly an "optimal" solution [1].

The general steps of an iterative procedure consists in constructing from a current solution *i* to the next solution *j* and in checking whether one should stop there or perform another step. Neighborhood search methods are iterative procedures in which a neighborhood *N(i)* is defined for each feasible solution *i*, and the next solution *j* is searched among the solutions in *N(i)* [2,3,4].

The origin of the Tabu Search (TS) went back to the 1970s and the modern form of TS was derived independently by Glover and Hansen [4,5]. The hybrids of the TS have

* Computer Sciences Department, University of Technology /Baghdad

https://doi.org/10.30684/etj.28.3.8
2412-0758/University of Technology-Iraq, Baghdad, Iraq

improved the quality of solutions in numerous areas such as scheduling, transportation, telecommunication, resource allocation, investment planning. The success of the TS method for solving optimization problems was due to its flexible memory structures which allowed the search to escape the trap of local optima and permitted to search the forbidden regions and explored regions thoroughly [2].

Generally, technique of backtracking means that the algorithm still keeps the previous visited (or generated) states (or nodes in graph) to recover these states as soon as the algorithm need their. Therefore, this important technique used in most heuristic search methods.

The outline of this paper is as follows. Section 2 describes the concepts of Tabu Search method with two basic algorithms. Section 3 deals with proposal of Backtracked Tabu Search (BTS) algorithm. Section 4 presents 2 case studies which are solved by BTS and TS with experimental results of each one. Section 5 includes the conclusions of this paper.

2- Tabu Search

Tabu Search (TS) is a meta-heuristic search which is designed to cross the boundaries of feasibility and search beyond the space of local optimality. The use of flexible memory based structures is the center strategy of the TS method [6]. While most exploration methods keep in memory essentially the value $f(i^*)$ of the best solution i^* visited so far, TS will also keep information on the itinerary through the last solution visited. Such information will be used to guide the move from i to next solution j to be chosen in $N(i)$. The role f the memory will be to restrict the choice of some subset of $N(i)$ by forbidding for instance moves to some neighbor solutions [7]. It would therefore be more appropriate to include TS in a class of procedures called dynamic neighborhood search techniques [6].

Formally let us consider an optimization problem in the following way : given a set S of feasible solutions and a function $f : S$ ® Â, find some solution i^* in S such that $f(i^*)$ is acceptable with respect to some criterion (or criteria). Generally a criterion of acceptability for a solution i^* would be to have $f(i^*) \leq f(i)$ for every i in S. In such situation TS would be an exact minimization algorithm provided the exploration process would guarantee that after a finite number of steps such an i^* would be reached [5,6].

In most contexts however no guarantee can be given that such an i^* will be obtained; therefore TS could simply be viewed as an extremely general heuristic procedure. Since TS will in fact include in its own operating rules some heuristic techniques, it would be more appropriate to characterize TS as a *metaheuristic*. Its role will often be to guide and to orient the search of another (more local) search procedure [7].

As a first step towards the description of TS, the classical descent method will be illustrated [1]:

Step 1: Choose an initial solution *i* in *S*.
Step 2: Generate a subset *V** of solution in *N(i)*.
Step 3: Find a best *j* in *V** (i.e. such that $f(i) \le f(k)$ for any *k* in *V**) and set *i* to *j*.
Step 4: If $f(j) \ge f(i)$ Then stop, Else go to Step 2.

In a straightforward descent method, we would generally take *V*=N(i)*. However this may often be too time-consuming: an appropriate choice of *V** may often be a substantial improvement.

Except for some special cases of convexity, the use of descent procedures is generally frustrating since the researchers are likely to be trapped in a local minimum which may be far (with respect to the value of *f*) from a global minimum [1,2].

As soon as non-improving moves are possible, the risk visiting again is a solution and more generally of cycling is presented. This is the point where the use of memory is helpful to forbid moves which might lead to recently visited solutions. If such memory is introduced we may consider that the structure of *N(i)* depend upon the itinerary and hence upon the iteration k; so we may refer to *N(i,k)* instead of *N(i)*. With these modifications in mind we may attempt to formalize an improvement of the descent algorithm in a way which will bring it closer to the general TS procedure. It could be stated as follows (*i** is the best solution found so far and *k* the iteration counter) [1,2]:

Step 1: Choose an initial solution *i* in *S*. Set *i*=i* and *k*=0.
Step 2: Set *k=k+1* and generate a subset *V** of solution in *N(i,k)*.
Step 3: Choose a best *j* in *V** (with respect to *f* or to some modified function $\tilde{f}$) and set *i = j*.
Step 4: If $f(i) < f(i^*)$ Then set $i^*=i$.
Step 5: If a stopping condition is met Then stop, Else go to Step 2.

Observe that the classical descent procedure is included in this formulation (the stopping rule would simply be $f(i) \ge f(i^*)$ and *i** would always be the last solution).

In TS some immediate stopping conditions could be the following [1,2,8]:

- $N(i,k+1)=\varnothing$.
- k is larger than the maximum number of iterations that allowed.
- the number of iterations since the last improvement of i* is larger than a specified number.
- evidence can be given than an optimum solution has been obtained.
- tabu list is full.
- no improved solutions.

While these stopping rules may have some influence on the search procedure and on its results, it is important to realize that the definition of *N(i,k)* at each iteration *k* and the choice of *V** are crucial [2].

The definition *N(i,k)* implies that some recently visited solutions are removed from *N(i)*; they are considered as tabu solutions which should be avoided in the next iteration. Such memory based on recent will partially prevent cycling. For instance keeping at iteration k a list T (tabu list) of the last |T| solutions visited will prevent cycles of size at most |T|. In such case $N(i,k)=N(i)$-T will be taken. However this list T may be extremely impractical in use; therefore the exploration process in S in terms of moves from one solution to the next [1,2]. In addition to, there are other versions of TS algorithms, but the above is the classical.

3- Proposal of Backtracked Tabu Search Algorithm

Generally, in the most heuristic search algorithms, the guarantee of finding the optimal solutions is the big problem. Also, local minimum (or maximum) represent the second big problem. Therefore, the heuristic search algorithms still in continuous developing. In this work, an attempt to improve the performance of TS using backtrack technique which permit to recover the best states when the system fail in the local minimum problem or it route the descent behavior far of the goal states.

Backtracking technique will adjust the behavior of TS by remain the other best neighbors in backtracked list (*B*) and back to select from this list whenever local minimum occurs or the best neighbor is not really best than the current state. In other words, any iterative exploration process should in some instance accept also non-improving moves from *i* to *j* in *V** (i.e. $f(j) > f(i)$) if one would like to escape from local minimum. Backtracking does this also, but it does not guide to choice of best j with storing the best neighbors, TS in contrast chooses a best j in V*, therefore the proposed version of TS will be more heuristic and robust to find the optimal solution or at least reduce the local minimum problem. The suggested BTS as following:

Step 1: Choose an initial solution *i* in *S*. Set $i^*=i$ and k=0.

Step 2: Set $k=k+1$ and generate a subset V^* of solution in $N(i,k)$.

Step 3: Choose a best *j* in V^* and set $i = j$.

Step 4: Select best subset from $N(i,k)$ add in *B*.

Step 5: If $f(i) < f(i^*)$ Then set $i^*=i$ Else choose the best subset in *B* and put in V^* and go to Step 3.

Step 6: Update *B*.

Step 7: If a stopping condition is met Then stop, Else go to Step 2.

where *B* represent the backtracked list which is contain the best neighbors of V^*, so the algorithm can recover the best previous states when the route of behavior far of the goal. The update step of *B* means delete the used neighbors and rearrange the others. In the next section illustrates the performance of BTS algorithm compare with others TS algorithms.

4- Case Studies & Experimental Results

Two standard optimization problems were used to test the proposal algorithm and to compare their performances with the original algorithm.

4-1 4-Color Map Problem

The celebrated 4 Color Map Theorem states that any map in the plane or on the sphere can be colored with only four colors such that no two neighboring countries are of the same color. The problem has a long history and inspired many people (including many non-mathematicians and in particular countless high school students) to attempt a solution [9].

The proof of the four color theorem by Haken and Appel [10] was so involved it required computational support to complete. It is well known that determining if a graph can be colored by a certain number of colors is NP-complete, but it is also known that even approximating the chromatic number of a graph is NP-hard [11]. There exist two main categories of algorithms: *successive augmentation algorithms* [12], which color a graph one vertex at a time, disallowing vertices from being re-colored and *iterative improvement algorithms*, which allow backtracking and re-coloring. Leighton's [13] RLF algorithm is an example of the first and Tabu searches and genetic algorithms are examples of the second [14].

In 4-color map problem there is a vector (N), where N is the number of cities in the map. An adjacency array of dimension NxN is used to identify the neighborhood of adjacent cities. The neighborhood search operator used is simply swapping two randomly chosen points.

4-2 Traveling Salesman Problem TSP

TSP is one of the major success stories for optimization because of its simplicity and applicability (or perhaps simply because of its intriguing name), the TSP has for decades served as an initial proving ground for new ideas related to both these alternatives. These new ideas make the TSP an ideal subject for a case study [15].

The origins of the Traveling Salesman Problem (TSP) are somewhat mysterious. It is a classical combinatorial optimization problem and can be described as follows: a salesman, who has to visit clients in different cities, wants to find the shortest path starting from his home city, visiting every city exactly once and ending back at the starting point. More formally [15]:

Given a set of n nodes and costs associated with each pair of nodes, find a closed tour of minimal total cost that contains every node exactly once.

In other words, a set $\{c_1, c_2, \ldots, c_N\}$ of *cities* is given and for each pair $\{c_i, c_j\}$ of distinct cities a *distance* $d(c_i, c_j)$. The goal is to find an ordering Π of the cities that minimizes the quantity

$$\sum_{i=1}^{N-1} d(c_{\Pi(i)}, c_{\Pi(i+1)}) + d(c_{\Pi(N)}, c_{\Pi(1)})$$

This quantity is referred to as the *tour length*, since it is the length of the tour a salesman would make when visiting the cities in the order specified by the permutation, returning at the end to the initial city. The concentrated in this paper would be on the *symmetric* TSP, in which the distances satisfy [15]:

$$d(c_i, c_j) = d(c_j, c_i) \text{ for } 1 \le i,\ j \le N$$

In computing terms the problem can be represented by a graph where all the nodes correspond to cities and the edges between nodes correspond to direct roads between cities [15].

In 4-color map problem there is

a vector (N), where N is the number of cities in the tour. An adjacency array of dimension NxN is used to identify the neighborhood of adjacent cities. The neighborhood search operator used is simply swapping two randomly chosen points.

4-3 Results

The researchers of TS have been proposed several modifications and hybrids algorithms with other techniques, one of these are Simulated Annealing Tabu Search (SATS) [16]. In this paper the proposed BTS will be compared with standard TS and SATS to illustrate the performance of each one.

In this paper, results of average 10 independent runs for all of these algorithms have proved that all of these algorithms are good technique capable of finding solutions close to the optimum, but a local minimum problem occur in very special cases. Results indicate that the proposal algorithm BTS have a faster convergence than the original TS and SATS.

Figure 1 illustrates the curve of number of iteration with number of cities in 4-color map problem in only solved cases using TS, SATS and BTS. Figure 2 illustrates the number of local minimum non-optimal solutions occur with number of cities in 4-color map problem using TS, SATS and BTS. Figure 3 illustrates the curve of number of iteration with number of cities in TSP in only solved cases using TS, SATS and BTS. Figure 4 illustrates the number of local minimum and non-optimal solutions occur with number of cities in TSP using TS, SATS and BTS.

5- Conclusions

The presented approach BTS is an important version of TS. BTS can increase the performance of optimal solutions finding, also, it can reduce the non-optimal solutions and local minimum problem. BTS depends on storing the best neighbors in backtracked list to recover them whenever the algorithm in local minimum or can not find the new best neighbor. The suggested approach achieves two important features of methods' searching which are called intensification and diversification. BTS gives less iteration numbers compare with TS and SATS. Also it has been reduced the non-optimal solutions and local minimum problem.

References

[1] A. Hertz, E. Taillard and D. de Werra, "*A Tutorial on Tabu Search*", EPFL, 1995.

[2] F. Glover, M. Laguna, A. Hertz, E. Taillard and D. de Werra, "*Tabu Search*", Annals of Operation Research, Vol. 41, 1993.

[3] F. Glover, "*Tabu Search, Part 2*", ORSA Journal on Computing 2, pp. 4-32, 1990.

[4] P. Hansen, and N. Mladenović, N., "*Variable Neighborhood Search: Principles and Applications*", European Journal of Operational Research, 130, pp. 449-467, 2001.

[5] F. Glover, "*Tabu Search, Part 1*", ORSA Journal on Computing 1, pp. 190-206, 1989.

[6] F. Glover and M. Leguna, "*Tabu Search*", Kluwer Academic Publisher, 1997.

[7] A. Hertz and D. de Werra, "*The Tabu Search Metaheuristic : how we used it*", Annals of Mathematics and Artificial Intelligence 1, pp. 111-121, 1990.

[8] D. Deng, J. Ma and H. Shen, "*A Simple and Efficient Tabu Search Heuristics for Kirkman Schoolgirl Problem*", Technical Report, University of Turku, Finland, 2005.

[9] Peter, Alfeld. "*Bivariate Splines and the Four Color Map Problem*", http://www.math.utah.edu/~alfeld/talks/S13/4CMP.html

[10] Wilson. Robin. "*Four Colors Suffice*", Princeton University Press, 2000.

[11] Garey. Johnson, D. S., *"Computers and Intractability: A Guide to the Theory of NP-Completeness"*, San Francisco: Freeman, 1977.

[12] Lewandowski, Gary. Condon, Anne. "*Experiments with Parallel Graph Coloring Heuristics and Applications of Graph Coloring"*. DIMACS Series in Discrete Mathematics, DIMACS ,1994.

[13] Leighton, F T. "*A Graph Coloring Algorithm for Large Scheduling Problems"*, Journal of Research of the National Bureau of Standards, Vol. 84, No. 6, pp 489-506, 1979.

[14] Palmer, Daniel. Kirschenbaum, Marc. Shifflet, Jason. Seiter, Linda. "*Swarm Reasoning"*, www.jcu.edu/math/swarm/papers/SIS2005.pdf

[15] Gaertner Dorian. "*Natural Algorithms for Optimisation Problems*". M.Sc. Thesis, Imperial College, 2004.

[16] A. Lim, B. Bodrigus and J. Zhang, "*Tabu Search Embedded Simulated Annealing for Shortest Route Cut and Fill Problem*", Journal of Operations Research Society, Vol. 56, No. 7, pp. 816-824, July 2005.

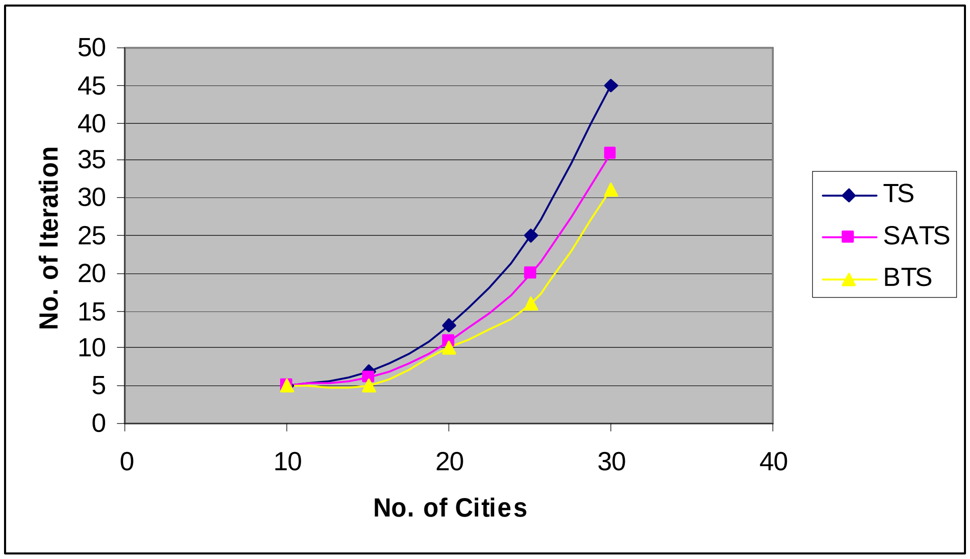

Figure (1) Average of No. of Iterations for 4-Color Map Problem Using TS, SATS and BTS

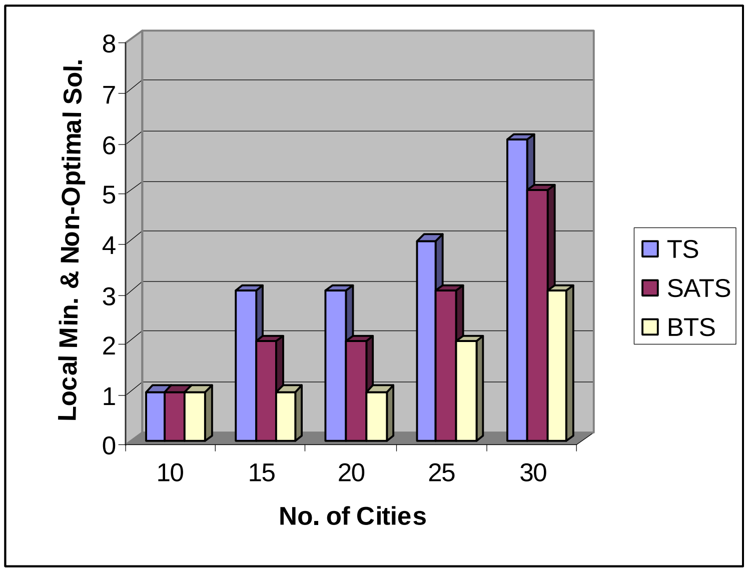

Figure (2) Average of Local Minimum and N-on-Optimal Solutions for 4-Color Map Problem Using TS, SATS and BTS

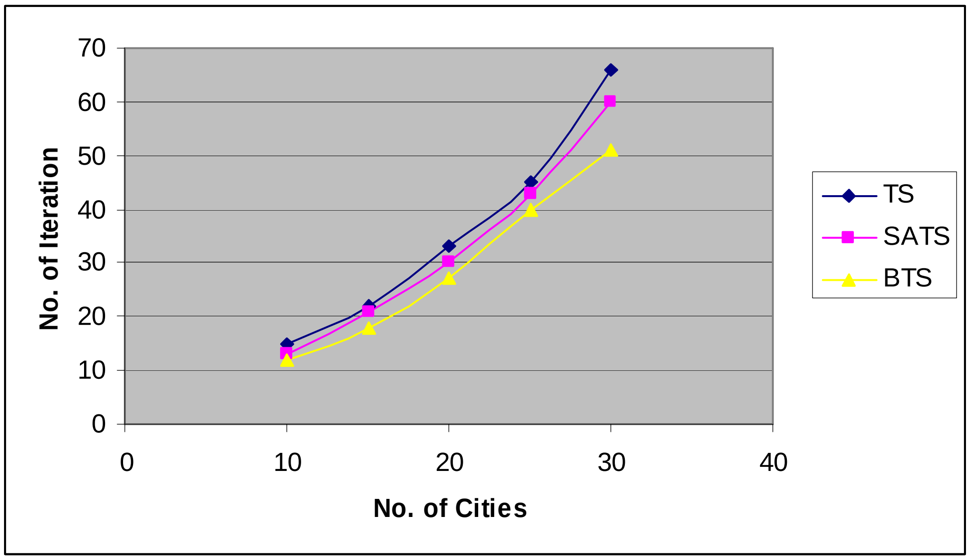

Figure (3) Average of No. of Iterations for TSP Using TS, SATS and BTS

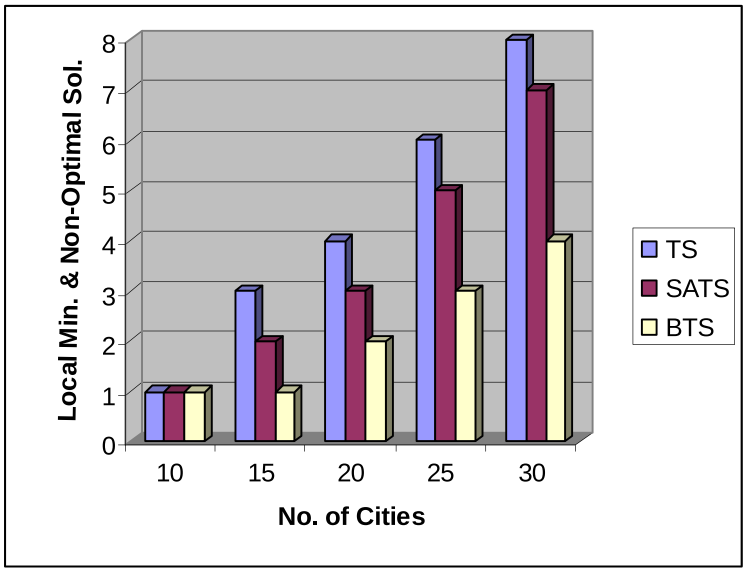

Figure (4) Average of Local Minimum and Non-Optimal Solutions for TSP Using TS, SATS and BTS