Proposal of Tabu Search Algorithm Based on Cuckoo Search

Ahmed T. Sadiq Al-Obaidi
Department of Computer Sciences
University of Technology
Baghdad, Iraq

Ahmed Badre Al-Deen Majeed
Quality Assurance Department
University of Baghdad
Baghdad, Iraq

***Abstract*—This paper presents a new version of Tabu Search (TS) based on Cuckoo Search (CS) called (Tabu-Cuckoo Search TCS) to reduce the effect of the TS problems. The proposed algorithm provides a more diversity to candidate solutions of TS. Two case studies have been solved using the proposed algorithm, 4-Color Map and Traveling Salesman Problem. The proposed algorithm gives a good result compare with the original, the iteration numbers are less and the local minimum or non-optimal solutions are less.**

Keywords—Tabu Search; Cuckoo Search; Heuristic Search; Neighborhood Search; Optimization; 4-Color Map; TSP

I. 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 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].

Cuckoo search was inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds (of other species). Some host birds can engage direct conflict with the intruding cuckoos. For example, if a host bird discovers the eggs are not their own, it will either throw these alien eggs away or simply abandon its nest and build a new nest elsewhere [7].

The objective of this paper is to improve the tabu search using the nature-inspired algorithm which cuckoo search. The outline of this paper is as follows. Section 2 describes the concepts of Tabu Search method with two basic algorithms. Section 3 includes the concepts of Cuckoo Search. Section 4 deals with proposal of Tabu-Cuckoo Search (TCS) algorithm. Section 4 presents 2 case studies which are solved by TCS and TS with experimental results of each one. Section 5 includes the conclusions of this paper.

II. 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 [7]. 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 [8]. It would therefore be more appropriate to include TS in a class of procedures called dynamic neighborhood search techniques [7].

Formally let us consider an optimization problem in the following way : given a set *S* of feasible solutions and a function $f : S \rightarrow \Re$, 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 [5,7].

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 [8].

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 $f(i) \leq f(k)$ for any *k* in *V**) and set *i* to *j*.
Step 4: If $f(j) \geq f(i)$ 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 *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 $f(i) \geq f(i^*)$ and *i** would always be the last solution).

In TS some immediate stopping conditions could be the following [1, 2, 9]:

- $N(i,k+1)=\varnothing$.
- 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.

III. Cuckoo Search

CS is a heuristic search algorithm which has been proposed recently by Yang and Deb [10]. The algorithm is inspired by the reproduction strategy of cuckoos. At the most basic level, cuckoos lay their eggs in the nests of other host birds, which may be of different species. The host bird may discover that the eggs are not its own and either destroy the egg or abandon the nest all together. This has resulted in the evolution of cuckoo eggs which mimic the eggs of local host birds. To apply this as an optimization tool, Yang and Deb used three ideal rules [10, 11]:

1) Each cuckoo lays one egg, which represents a set of solution co-ordinates, at a time and dumps it in a random nest;

2) A fraction of the nests containing the best eggs, or solutions, will carry over to the next generation;

3) The number of nests is fixed and there is a probability that a host can discover an alien egg. If this happens, the host can either discard the egg or the nest and this result in building a new nest in a new location. Based on these three rules, the basic steps of the Cuckoo Search (CS) can be summarized as the pseudo code shown as below [10, 11, 12].

Cuckoo Search via Levy Flight Algorithm

```
Input: Population of the problem;
Output: The best of solutions;
  Objective function f(x), $x = (x_1, x_2, ...x_d)^T$
  Generate initial population of n host nests $x_i$
                                    (i = 1, 2, ..., n)
  While (t <Max Generation) or (stop criterion)
      Get a cuckoo randomly by Levy flight
      Evaluate its quality/fitness $F_i$
      Choose a nest among n(say,j)randomly
      If ($F_i > F_j$) replace j by the new solution;
      A fraction(pa) of worse nests are abandoned and new
        ones are built;
      Keep the best solutions (or nests with quality
        solutions);
      Rank the solutions and find the current best;
      Pass the current best solutions to the next generation;
  End While
```

When generating new solution $x^{(t+1)}$ for, say cuckoo *i*, a Levy flight is performed

$$x^{(t+1)}_i = x(t)_i + \alpha \oplus Levy(\beta) \text{ ……. } (1)$$

where $\alpha > 0$ is the step size which should be related to the scales of the problem of interests. In most cases, we can use $\alpha = 1$. The product $\oplus$ means entry-wise walk while multiplications. Levy flights essentially provide a random walk while their random steps are drawn from a Levy Distribution for large steps

$$Levy \sim u = t^{-1-\beta} \quad (0 < \beta \leq 2) \ldots\ldots\ldots (2)$$

this has an infinite variance with an infinite mean. Here the consecutive jumps/steps of a cuckoo essentially form a random walk process which obeys a power-law step-length distribution with a heavy tail. In addition, a fraction *pa* of the worst nests can be abandoned so that new nests can be built at new locations by random walks and mixing. The mixing of the eggs/solutions can be performed by random permutation according to the similarity/difference to the host eggs.

IV. Proposal of Tabu Search Algorithm Based on Cuckoo Search

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 CS which is provides more diversity to candidate solutions of TS. CS will call in the TS when there are no more good solutions in TS. Initially, CS will be work with best solutions list (*B*) and replace the old solutions of tabu list by the CS solutions to provide a good diversity to TS candidate solutions. 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, CS does this. 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 TCS 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 there is no best solution Then call the Cuckoo Search with best subset from Tabu List.
Step 6: Select the best solutions from Cuckoo Search output to add in the Tabu List.
Step 7: If a stopping condition is met Then stop, Else go to Step 2.

where *B* represent the currently best solutions 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 TCS algorithm compare with others TS algorithms.

V. Case Studies and Experimental Results

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

A. 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 [13].

The proof of the four color theorem by Haken and Appel [14] 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 [15]. There exist two main categories of algorithms: *successive augmentation algorithms* [16], 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 [17] RLF algorithm is an example of the first and Tabu searches and genetic algorithms are examples of the second [18].

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.

B. 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 [19].

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 [19]:

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 [19]:

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

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 [19].

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.

C. 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) [20]. In this paper the proposed TCS 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 TCS 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 TCS. 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 TCS. Figure 3 illustrates the curve of number of iteration with number of cities in TSP in only solved cases using TS, SATS and TCS. Figure 4 illustrates the number of local minimum and non-optimal solutions occur with number of cities in TSP using TS, SATS and TCS.

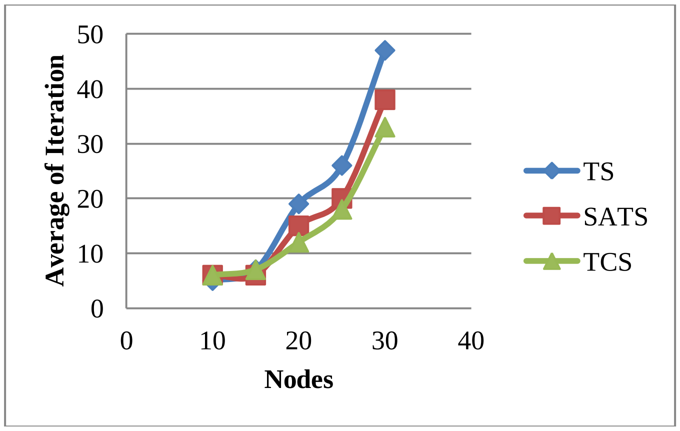

Fig. 1. Average of No. of Iterations for 4-Color Map Problem Using TS, SATS and TCS

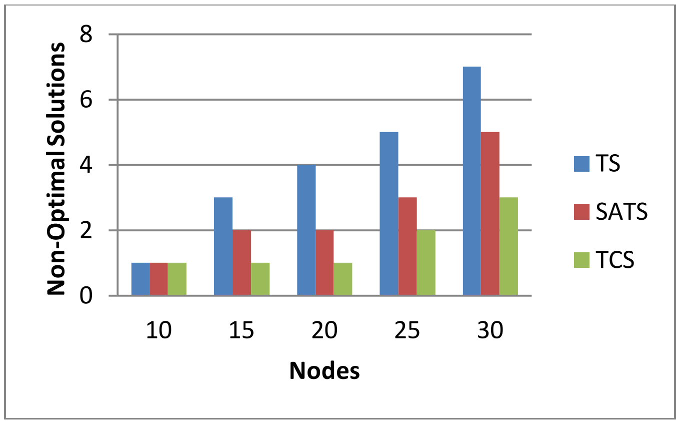

Fig. 2. Average of Non-Optimal Solutions for 4-Color Map Problem Using TS, SATS and TCS

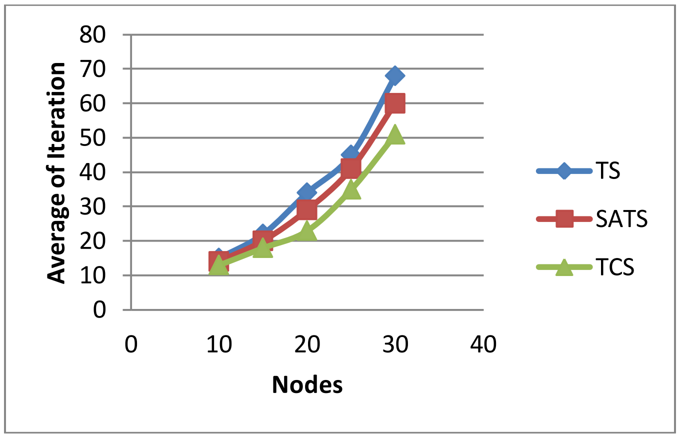

Fig. 3. Average of No. of Iterations for TSP Using TS, SATS and TCS

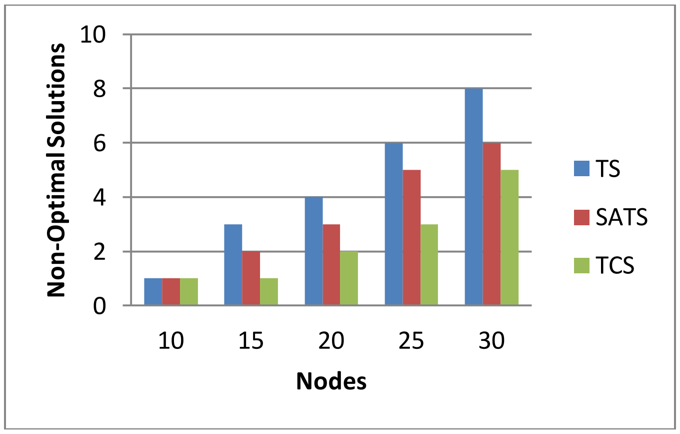

Fig. 4. Average of Non-Optimal Solutions for TSP Using TS, SATS and TCS

VI. CONCLUSIONS

The presented approach TCS is an important version of TS. TCS can increase the performance of optimal solutions finding, also, it can reduce the non-optimal solutions and local minimum problem. TCS depends on storing the best neighbors in the currently best solutions list to use these solutions in the CS to for improving whenever the algorithm in local minimum or cannot find the new best neighbor. The suggested approach achieves two important features of methods' searching which are called intensification and diversification. TCS 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] R. B. Payne, M. D. Sorenson, and K. Klitz, "*The Cuckoos*", Oxford University Press, (2005).

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

[8] 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.

[9] 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.

[10] X. S. Yang and S. Deb, "*Cuckoo Search via Lévy Flights*". World Congress on Nature & Biologically Inspired Computing (NaBIC 2009). IEEE Publications. pp. 210–214, December, 2009.

[11] H. Zheng and Y. Zhou, "*A Novel Cuckoo Search Optimization Algorithm Base on Gauss Distribution*", Journal of Computational Information Systems 8: 10, 4193–4200, 2012.

[12] Xin-She Yang, "*Cuckoo Search and Firefly Algorithm*", Springer Press, 2014.

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

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

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

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

[17] 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.

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

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

[20] 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.