Analysis of neighborhood generation and move selection strategies on the performance of Tabu Search

FARHAD KOLAHAN[1], AHMAD TAVAKOLI [2], BEHRANG TAJDIN, MODJTABA HOSAYNI
[1]Mechanical Engineering Department, [2]Department of Management
Ferdowsi University of Mashhad
IRAN

Abstract: - In recent years, heuristic algorithms are widely used to solve very large and complicated optimization problems. In this paper, the structure of Tabu Search (TS) - one of the most efficient neighborhood search algorithms - is fist explained. Then the effect of several neighborhood generation and move selection mechanisms on its performance is investigated. To reach this goal, a set of constrained Traveling Salesman Problems (TSP) is solved by five different neighborhoods. The performance of TS algorithm is then evaluated and compared from the view point of convergence speed and solutions qualities.

Key-Words: Optimization algorithms, Tabu Search, neighborhood generation, move strategies, TSP

1 Introduction

Nowadays, modern industries are faced with different issues, such as severe competition, rapid changes, energy and resource limitations, environmental concerns and so on. Because of these challenges, process optimization - that in necessary for efficient usage of resources and increasing the operating efficiency- has attracted a lot of attention by researchers. However, real sized optimization problems are mostly large and complicated. Therefore, solving such problems by traditional deterministic methods is very time consuming and inefficient. On the other hand, simplifying complicated and large problems usually makes their results inaccurate and unsuitable for real life situations. Heuristic algorithms such as Tabu Search and Genetic Algorithm, with high calculation speed and good (not necessarily optimum) solution quality, seem to be good alternatives for deterministic optimization procedures. One of the important benefits of these algorithms is their good adoption to broad band of optimization problems. These methods are working within feasible solutions space and are not dependent on the problem structure. Thus, as long as the decision variables are discrete or they can be defined as discrete variables, complicated problems can be coded and solved by these methods. This is done by converting each solution into a numerical string of decision variables. However, even after adjusting the algorithm on problem structure, selected values for parameters have a great effect on calculation speed and the quality of final solutions.

In some research papers, the performances of some neighborhood search heuristics are compared using different standard problems such as TSPs. Nevertheless, such comparisons may not be fair and exact, since with proper search parameter settings most techniques may provide good answers. In other words, the performance of any algorithm is affected by the way that its parameters have been tuned.

This paper has a different approach. Our aim is to evaluate the effects of various neighborhood generation mechanisms and move selections policies on the performance of Tabu Search (TS). To achieve this goal, first different neighborhood generation mechanisms and move selection policies in TS are presented. Then, using a set of TSP numerical examples, their effects on algorithm performance is investigated. Finally, search results are compared and discussed in terms of computational speed and solutions qualities.

2 Tabu Search

Since the mid 80s, with the advent in computer calculation capabilities, many neighborhood search algorithms have been proposed to solve large and complicated optimization problems. Generally, most of these heuristics are inspired by the natural and physical phenomena. To some extend, they are a simulation of natural phenomenon by mathematical functions. These methods do not guarantee optimum solutions but by searching the feasible solution space try to find an optimum or near optimum solution for the problem in hand. That is why they are called "neighborhood search methods". Genetic

Algorithm (GA), Simulated Annealing (SA) and Tabu Search (TS) are some well known such algorithms.
Tabu Search, first proposed by Glover [1], is an iterative neighborhood search that step by step, searches feasible solution space to find the optimum or near optimum solution. In this method, search begins from a feasible solution and for each move, the neighborhood of current solution, is generated and evaluated. Then a new move is made to the best allowable (non-tabu) answer in the neighborhood. The stepwise transition from one solution to another allows the search to reach an optimal or a close-to-optimal solution after a number of moves. However, a single move, by itself, may not necessarily improve the current value of objective function. This distinguishes tabu search from other traditional techniques such as hill climbing that require each move to be an improving step. Throughout the search, the best solution found so far, *Cbest*, and its corresponding sequence, *Sbest*, will be recorded and updated. The important parameters in TS are as follows:
Starting Point: Search begins from a feasible solution such as ***S***. A feasible solution is any set of values for decision variables that satisfies problem specifications and constraints.
Neighborhood: For a given solution *S*, the neighborhood *N(S)* is a set of feasible solution generated with the minimum changes in current solution. Pairwise interchange is the most popular mechanism for neighborhood generation. In this mechanism, each neighbor is generated by changing the position of two members in the current solution string.
Move: A move is the transition from the best solution, s^{**}, in the previous neighborhood to the best permissible solution, s^{*}, that has the best cost function value, in the current neighborhood. Such a move may or may not be an improving one.
Tabu List: One of the important features of tabu search is its ability to avoid being trapped in local optima by constructing a list of tabu moves. Tabu list, *T_list*, includes a certain number (*T_siz*e) of previous moves which are not allowed at the current iteration. Once a move from s^{**} to s^{*} is made, s^{**} is added to the top of tabu list and the oldest member of the list is removed. Thus, returning back to this s^{**} is forbidden for the next T_size iterations. This can exclude, to some extent, those moves which lead to possible cycling. The size of tabu list can affect the search performance. Although a longer list may prevent cycling, it requires more scanning and may limit the search domain. The best tabu list size appears to be problem dependent and there is no fixed rule to follow in determining tabu-list size so far.
Termination Criteria: The last element necessary for tabu search is termination criterion. In general, search can be stopped when a certain number of iterations, Mmax, is completed, after a pre-defined of computational time, Tmax, is reached, or when no improvement can be obtained in a specific number of moves.
A full explanation of this technique and some of its applications can be found in [2,3,4].

3 Neighborhood Generation and Move Selection Mechanisms

Generally, parameters setting have a considerable impact on in heuristic algorithms performances. In many cases, it takes many trial runs to find suitable values for search parameters. This, in turn, makes optimization process a time consuming and demanding task. Fortunately, in comparison to the similar methods, TS is a robust algorithm with few parameters.
Neighborhood generation mechanism and move selection policy are the most important parameters that affect TS performance in terms of solution quality and computational time. Conventionally, a neighborhood, N(S), is defined as a set of solutions that can be obtained by performing one transition in the current solution. In the related literature, pairwise interchange is the most widely used technique to make such a transition. In this method, a solution is generated by switching the members (cities in our case) in positions i and j. The complete pairwise interchanges of a J-city problem leads to [N(s)] = J(J - 1)/2 neighbors. Extraction and reinsertion is another technique for transition. With this method, the neighborhood contains all solutions obtained by extracting the city in position i and inserting it right after (or before) the city in position j. The neighborhood size for the extraction and reinsertion approach is [N(s)] = J(J - 1)2 which is almost doubled as compared to pairwise interchange. Thus this mechanism appears to be more computationally demanding. Furthermore, as indicated by Adenso-Diaz [5], none of these two techniques seems to outperform the other in terms of solution quality for a given run time. Therefore, in this paper only the pairwise interchange approach is used as the basic neighborhood generation mechanism.
The next step in tabu search is to specify a move strategy. The classic approach is to evaluate the entire neighborhood and choose the best allowable

move. However, the required computational time could be unacceptably long when the problem size is large. To overcome this problem, partial search schemes have been proposed recently [6].

In following, five different neighborhood generation and move selection mechanisms are discussed. These policies are then evaluated using a numerical example of 100 cities.

Pairwise Interchange and evaluation of all neighborhood (PI): This is the most common police in which all neighbors of the current solution are generated and evaluated at each iteration. For a problem with J variables (cities), the transposition range (the range of interchange) would be from 1 to $(J-1)$. The neighborhood size for PI policy is $|N(s)| = J(J-1)/2$. This number of solutions should be generated and evaluated for each move. This policy guarantees complete neighborhood search and results in a better solution quality. Nevertheless, as problem size grows, generation and evaluation of all neighbors can be very time consuming and would cause sluggish convergence. To increase the search speed, partial neighborhood can be generated and evaluated.

Partial Neighborhood (PN): The range of interchange in PN is from 1 to m (m < (J-1)). In other word, neighborhood is generated by interchanging the position of each city with a limited number of next cities. Indeed, this will lead to a smaller neighborhood and will increase search speed. A problem with J city has $|N(s)| = m.(J-m) + \sum_{i=1}^{m}(i-1)$ number of neighbors.

Random Neighborhood (RN): In this method, m cities from J-1 possible cities are selected randomly and interchange will be done on them. This mechanism is similar to PN in terms of neighborhood size, but has random nature.

Adjacent Pairwise Interchange (API): This is a special case of PN method, in which m has been set to 1; i.e. transposition range is only 1. This mechanism generates the smallest neighborhood size. The number of neighbors in each move is $|N(s)| = (J-1)$.

First Improving Neighborhood (FIN): The mechanism of neighborhood generation is the same as PI, but move would be made to the first neighbor that improves objective function. In FIN policy, neighbors for current solution are generated one at the time and once an improving neighbor is found the process of neighborhood generation will stop. If no improving neighbor is found, the entire neighborhood is generated and move is made to the best one; just like PI. Therefore, the size of neighborhood is unpredictable and can vary from 1 to $|N(s)| = J(J-1)/2$.

4 Tabu Search for TSP

Traveling Salesman Problem is a classical optimization problem in which a salesperson has to visit all interconnected cities only once in such way that the total traveling distance is minimized [7]. In constrained TSP, there is no direct connection between some cities. This is a more practical version of TSP that can be implemented to many optimization problems [8]; such as job sequencing [2], facility location [9] and vehicle routing [10]. It is, therefore, considered in this research as the benchmark problem. A schematic example of constrained TSP is shown in Figure 1 in which all 5 nodes are interconnected but there is no direct connection between nodes C and D. Each solution (tour) is represented by a sequence of cities such as A,B,C,E,D. For a problem of size *J*, there is a total of *J!* possible sequences or tours. The cost function is the sum of total crossed distances in a given tour. It should be noted that in constrained TSP some tours may be infeasible due to the absence of direct connection between two consecutive cities (e.g. A,B,C,D,E,).

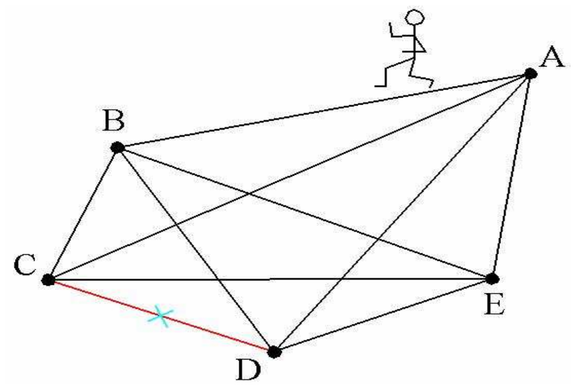

Fig 1. Schematic representation of constrained TSP

To solve TSP by Tabu Search, each solution is represented by a string that shows the sequence of cities in a unique way. The search starts with a feasible tour or sequence, say *S*. Then the neighborhood *N(S)*, is generated by performing one transition in the current solution; i.e. pairwise interchange. Each neighbor in *N(s)* has its associated objective function value or total distance *G(s),* and the one with the smallest *G(s)* is defined as the best neighbor, denoted by s^*. A move is then made from s^{**}, the best sequence of the immediate previous neighborhood, to s^*, provided that s^* is not in the

current tabu list. The best solution found so far, and its corresponding sequence are then updated if necessary and kept in memory. The sequence s^* is then stacked into the tabu list of pre-defined size and the oldest sequence is removed from the list. The search is stopped when termination criterion is met.
The size of neighborhood and the way that the next move is selected has a great effect on search performance. This becomes more evidence when the problem size is large.

5 Numerical Example and Results

In order to analyze different neighborhood generation and move selection mechanisms, the TS algorithm has been applied to a set of TSP problems with 100 cities (variables). In TSP problem, the traveling salesperson must visit all cities with minimum traveling distance. To make the optimization problem more realistic, we use constrained problem in which there are no direct connections among some cities. The algorithm has been coded on MATLAB R6.5 software. For more accurate and fair comparison of results, in all runs the arrangement of cities, the termination criterion and the starting points are kept the same.

Computational results and convergence curves for a sample problem are presented in Table 1 and Figure 2 respectively.

Table 1. Results of TSP example with 100 cities

Neighborhood Mechanism	Initial distance	Final distance	Objective improvement (%)	Neighborhood size $\lvert N(s) \rvert$
PI	42743	16160	164.5	4950
RN		16120	165.1	2535
PN		16980	151.7	2535
API		25550	67.3	99
FIN		16020	166.8	~
DN		16160	164.5	~

The results in Table 1 show considerable objective function improvements for all policies during 30 seconds of search time. The least cost function (distance) improvement is more than 67% while there is about 167% reduction for the best scenario. Nevertheless, there is a big difference in terms of search performance under different policies. This shows the importance of neighborhood generation and move selection policies on the search effectiveness.

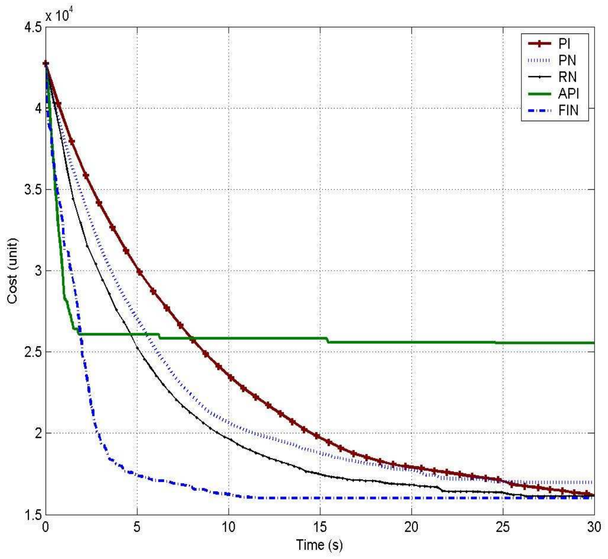

Fig. 2 Convergence curves for TSP problem

As shown in Figure 2, all mechanisms (excluding API) converge to the same answer. Although API neighborhood demonstrates a fast convergence rate at the beginning, it performs inferior in terms of solution quality. The fast convergence rate of API is a result of its small neighborhood size; whereas the same feature probably causes it being trapped in local optima.

Among four other mechanisms, FIN method seems to be the best as it demonstrates both superior convergence rate and solution quality. Its dynamic neighborhood size allows the algorithm to move fast towards promising parts of solution space. It is also capable of doing a complete neighborhood search at the final stages where possible optimum is near by. This reduces the computational time at the beginning of the search while improves solution quality at final iterations.

Other three mechanisms have somehow similar patterns. Relative dullness of PI mechanism is because of its neighborhood size and move selection policy that requires generating and evaluation of all neighbor solution for each move. On the other hand, evaluating all neighbors will increase the probability of finding an optimum solution. Although in the example problem the neighborhood sizes of RN and PN are the same, RN clearly outperforms PN. The random nature of search, which extends the search area, in RN is the main reason for this advantage. This is somehow similar to what happens in Simulated Annealing, another powerful probabilistic neighborhood search.

6 Conclusion

Tabu Search algorithm is an efficient optimization techniques. With its few search parameters, it is a robust algorithm with little required tuning. In this research, the effects of different neighborhood generation and evaluation mechanisms on search efficiency were analyzed. Computational results have shown that in the longer run times, most neighborhood mechanisms result in similar solutions. However, the rate of improvements largely depends on the type of neighborhood. This becomes more evident for shorter computational times and for larger problem sizes. It seems dynamic neighborhoods such as FIN lead to better solutions in shorter search times. This is mostly because of the part of neighborhood to be evaluated varies in different phases of the search. In the beginning, smaller neighborhood is usually evaluated allowing the search to expand search domain. After several iterations when the algorithm converges to promising part of solution space, bigger neighborhoods are generated and evaluated to increase solution quality. Therefore, it may be beneficial to use different neighborhood mechanisms at different stages of the search.

Although we can not extend the results find here to all optimization problems, it is clearly shown that the performance of Tabu Search, and any other neighborhood search, is greatly affected by its neighborhood generation and move selection mechanisms. This could be a promising area for future research works.

References:

[1] F. Glover, Tabu Search – a tutorial, *Interfaces*, Vol. 20, 1990, pp 74-94.

[2] F. Kolahan, M.Liang, An adaptive TS approach JIT sequencing with variable processing times and sequence-dependent setups, *European Journal of Operations Research*, No. 109, 1998, pp. 142-159.

[3] D.L. Woodruff, Simulated annealing and tabu search: Lessons from a line search, *Computers and Operations Research*, Vol. 41, No. 8, 1994, pp. 829-831.

[4] A. Hertz, D. De Werra, The tabu search *metaheurestic*: how we used it, *Mathematics in Artificial Intelligence*, Vol. 1, 1991, pp 111–121.

[5] B. Adenso-Diaz, Restricted neighborhood in the tabu *search* for the flowshop problem, *European Journal of Operational Research,* Vol. 62, 1992, pp. 27-37.

[6] F. Della Croce, Generalized pairwise interchanges and machine scheduling. *European Journal of Operations Research*, Vol. 83, 1995, pp. 310-319.

[7] B. Golden, L. Bodin, and T. Doyle, Approximation traveling salesman algorithm, *Operations Research*, No. 28, 1980, pp 694-712.

[8] K. Katayama, H. Sakamoto and H. Narihisa, The efficiency of hybrid mutation genetic algorithm for the travelling salesman problem, *Mathematical and Computer Modeling*, Vol. 31, No.12, 2000, pp. 197-203.

[9] Y. Chan and S.F. Baker, The multiple depot, multiple traveling salesmen facility-location problem: Vehicle range, service frequency, and heuristic implementations, *Mathematical and Computer Modeling*, Vol. 41, 2005, pp. 1035-1053

[10] M. Jin, K. Liu and R. O. Bowden, A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem, *International Journal of Production Economics*, Vol. 105, No. 1, 2007, pp. 228-242