DESIGN AND IMPLEMENTATION OF TRAVELLING SALESMAN PROBLEM USING TABU SEARCH ALGORITHM

Nishita. A. Thakkar[1], Poonam. S. Durge[2], Mrunal A. Munje[3], Mona R. Choutmal[4]
COMPUTER TECHNOLOGY
mrunalmunje28@gmail.com,poonamdurge03@gmail.com,nishitathakkar.mid@gmail.com monachoutmal12@gmail.com

***Abstract*-This paper discusses the implementation of a robust tabu search algorithm for travelling salesman problem. It explores the meta heuristic approach called tabu search, which is dramatically changing our ability to solve a host of problems in applied science, business and engineering.**

Keywords: Optimization, NP-hard ,Tabu Search,

Introduction

Engineering and technology have been continuously providing examples of difficult optimization problems. In this talk we shall present the tabu search technique which with its various ingredients may be viewed as an engineer designed approach: no clean proof of convergence is known but the technique has shown a remarkable efficiency on many problems. The roots of tabu search go back to the 1970's; it was first presented in its present form by Glover [Glover, 1986]; the basic ideas have also been sketched by Hansen [Hansen 1986]. Additional efforts of formalization are reported in [Glover, 1989], [de Werra & Hertz, 1989], [Glover, 1990]. Many computational experiments have shown that tabu search has now become an established optimization technique which can compete with almost all known techniques and which - by its flexibility - can beat many classical procedures. Up to now, there is no formal explanation of this good behaviour. Recently, theoretical aspects of tabu search have been investigated .
The most famous neighbourhood search method which has been used for finding an approximation to the minimum value of a real-valued function *f* on a set *S* is the descent method. It is outlined below :

Descent method
Step 1. Choose an initial solution *i* in *S*.
Step 2. Find a best *j* in *N(i)* (i.e. such that *f(j)* £ *f(k)* for any *k* in *N(i)*).

Step 3. If *f(j)* ³ *f(i)* then stop. Else set *i=j* and go to Step 2.
Such a method clearly may stop at a local but not global minimum of *f*. In general, *N(i)* is not defined explicitely: we may search for *j* by exploring some directions from *i* (for instance the coordinate axes). Simulated annealing and tabu search can be considered as neighbourhood search methods which are more elaborate than the descent method. The basic ingredients of tabu search are described in the next section.

Literature review

H. S. Abdinnour and S. W. Hadley, "Tabu search based heuristics for multi-floor facility layout", *International Journal of Production Research*, shows how tabu how tabu search is implemented. This tabu search can be used for implementing travelling salesman problem to generate the best possible path for the salesman to reach from initial to final node.

Basic ideas of Tabu Search

In order to improve the efficiency of the exploration process, one needs to keep track not only of local information (like the current value of the objective function) but also of some information related to the exploration process. This systematic use of *memory* is an essential feature of tabu search (TS). Its role will be emphasized later .

Definitions and terminologies

Tabu Search (TS) was developed by Fred Glover in 1988. It was initiated as an alternative local search algorithm addressing combinatorial optimization problems in many fields like scheduling, computer channel balancing, cluster analysis, space planning etc. (Glover, 1989). However, popularization and dissemination of TS goes back to the works of Hertz and de Werra (1987, 1989, 1991). This section consists of three parts: general definitions, TS related definitions, and definitions related to other meta-heuristics.

Basic ideas of Tabu Search

In order to improve the efficiency of the exploration process, one needs to keep track not only of local information (like the current value of the objective function) but also of some information related to the exploration process. This systematic use of *memory* is an essential feature of tabu search (TS). Its role will be emphasized later on. 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 solutions visited. Such information will be used to guide the move from *i* to the next solution *j* to be chosen in *N(i)*. The role of the memory will be to restrict the choice to some subset of *N(i)* by forbidding for instance moves to some neighbour solutions.

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 willin fact include in its own operating rules some heuristic techniques, it would be more appropriate to characterize TS as a *metaheuristic*. Its role will most often be to guide and to orient the search of another (more local) search procedure.

As a first step towards the description of TS, we reformulate the classical descent method as follows:

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(j)* *f(k)* for any *k* in *V**) and set *i=j*.

Step 4. If *f(j)* *f(i)* then stop. Else go to Step 2.

In a straightforward descent method we would generally take *V*=N(I* to the value of *f*) from a global minimum. So any iterative exploration process should in some instances accept also non-improving moves from *i* to *j* in *V** (i.e. with *f(j)* > *f(i)*) if one would like to escape from a local minimum.

Observe that the classical descent procedure is included in this formulation . Notice also that we may consider the use of a modified *f* instead of *f* in some circumstances to be described later.

In TS some immediate stopping conditions could be the following:

- k is larger than the maximum number of iterations 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.

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.. We now have described almost all basic ingredients of TS. There is one additional feature which is important in the exploration process; it is related to the fact that in the process *f* may in some instances be replaced by another function .This will allow us to introduce some intensification and diversification of the search. In TS memory is used in different ways to guide the search procedure; we have seen a short term memory whose role was to forbid some moves likely to drive us back to recently visited solutions. Memory is also present at a deeper level.

In the search process it is sometimes fruitful to intensify the search in some region of S because it may contain some acceptable solutions.

Variable tabu list size

We have seen that it may be convenient to use several tabu lists at a time. We will restrict the discussion to the case of a unique tabu list but the following may be extended to the general case. The basic role of the tabu list is to prevent cycling. If the length of the list is too small, this role might not be achieved; conversely a too long size creates too many restrictions and it has been observed that the mean value of the visited solutions grows with the increase of the tabu list size. Usually, an order of magnitude of this size may be easily determined. However, given an optimization problem it is often difficult or even impossible to find a value that prevents cycling and does not excessively restrict the search for all instances of the problem of a given size. An effective way for circumventing this difficulty is to use a tabu list with variable size. Each element of the list belongs to it for a number of iterations that is bounded by given maximal and minimal values.

Tabu Search (TS)

TS can be considered as a generalization of iterative improvements like SA. It is regarded as an adaptive procedure having the ability to use many methods, such as linear programming algorithms and specialized heuristics, which it guides to overcome the limitations of local optimality (Glover, 1989).

TS is based on concepts that can be used in both artificial intelligence and optimization fields. Over the years TS was improved by many researchers to become one of the preferred solution approaches. Surrogate constraints, cutting plane approaches, and steepest ascent are big milestones in the improvement of TS. TS applies restrictions to guide the search to diverse regions. TS has memory property that distinguishes it from other search designs. It has adaptive memory that is also different from rigid memory used by branch and bound strategies. Memory in TS has four dimensions: quality, recency, frequency, and influence. TS forces a move to a neighbor with least cost deterioration. TS uses memory to keep track of solutions previously visited so that it can prevent revisiting that solution. Memory-based strategies are hallmark of TS approaches. Many applications don't include advanced features of TS since good solutions

are typically achieved by simple designs. A basic tabu search algorithm for a maximization problem is illustrated in Figure

```
algorithm Tabu search
  begin
        T:= [ ];
        s:=initial solution;
       s*:=s
    repeat
        find the best admissible s' ϵ N(s);
        if f(s') > f(s*) then s*:=s'
        s:=s';
        update tabu list T;
     until stopping criterion:
 end;
```

Figure– A basic tabu search algorithm

Tabu list

The simplest way such a memory can be used is to remember the last *k* solutions that have been visited, and to avoid these solutions. That is, the algorithm keeps a *list* of tabu solutions. The length of this *tabu list* may be varied, and a longer list will prevent cycles of greater length – a list of length *k* will prevent cycles of length ≤ *k*. However, it may be impractical (e.g. time consuming to compare the solutions) to incorporate such memory. As such, *simpler* type of memories is usually incorporated, one of which is remembering the
last *k moves* made. A move for a solution *si* can viewed as 'changes' applied to the solution *si*, to get to a new solution *si*+1 ∈ *N*(*si*). Generally, moves are defined so that they are *reversible*: for a move *m* from *si* to *si*+1, there is a *reverse* move *m*-1 from *si*+1 to *si*. The tabu search can then keep a tabu list of the last *k* reverse moves made, and avoid making these moves To partially overcome the restriction imposed by using a move list (for the tabu list), a tabu search usually incorporates *aspiration conditions*, whereby a tabu move (a move that is in the tabu list) is selected if it satisfies one or more of the aspiration conditions. One simple aspiration condition that is commonly used in tabu search is the following.
If a tabu move leads to a solution that is better than the best solution found so far, then it should be selected.
Finally, one other important feature of tabu search is that of *search intensification* and *search diversification*. For this purpose, a more elaborate memory structure that takes into account the *recency*, *frequency*, and *quality* of solutions and moves made so far is used.

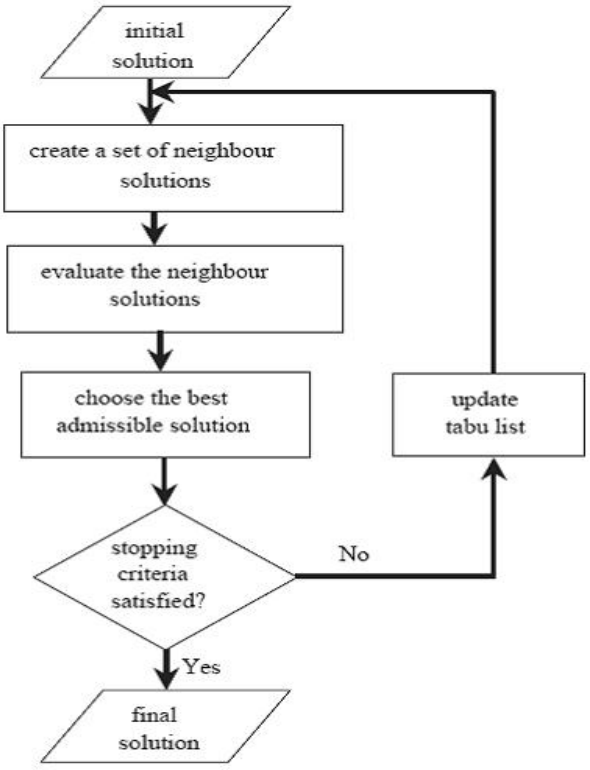
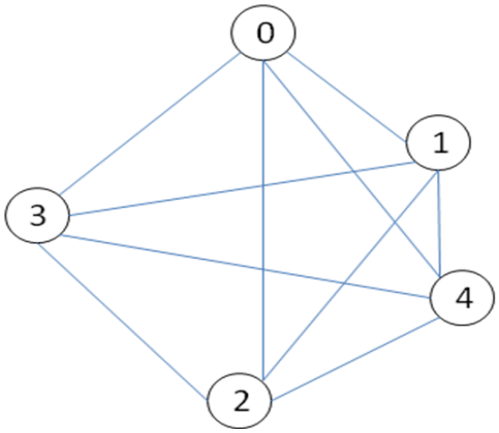


Fig. 2. Flowchart of a standard TS algorithm.

Implementation of Tabu search

Travelling salesman problem:



	0	1	2	3	4
0	**0**	**2**	**5**	**1**	**3**
1	**2**	**0**	**5**	**3**	**4**
2	**5**	**1**	**0**	**3**	**2**

3	**1**	**3**	**3**	**0**	**4**
4	**3**	**3**	**2**	**4**	**0**

Experimental results:

Initial Solution : {0,2,3,4,1,0}
I : Cost = 5+3+4+3+2 = 17
Now current best cost is 17.
Initial Solution : {0,2,3,4,1,0}
Now to find the new best cost Swap the cities
ex. (1,2) (1,3) (1,4) (2,3) (2,4) (3,4)

1. (1,2)
   Initial Sol. {0,2,3,4,1,0}
   After Swapping { 0,3,2,4,1,0}
   Cost : 1+3+2+3+2 =11
   Best Cost = 11
2. (1,3) Initial Sol. { 0,3,2,4,1,0}
   After Swapping {0,4,2,3,1,0}
   Cost : 3+2+3+2+2 = 12
3. (1,4) Initial Sol. { 0,3,2,4,1,0}
   After Swapping {0,1,2,4,3,0}
   Cost : 2+1+2+4+1 = 10
   Best Cost = 10
4. (2,3) Initial Sol. {0,1,2,4,3,0}
   After Swapping {0,1,4,2,3,0}
   Cost = 2+3+2+3+1 =11
5. (2,4) Initial Sol. {0,1,2,4,3,0}
   After Swapping {0,1,3,4,2,0}
   Cost = 2+2+4+2+5 = 15
6. (3,4) Initial Sol. {0,1,2,4,3,0}
   After Swapping {0,1,2,3,4,0}
   Cost = 2+1+3+4+3 = 13

So after first iteration Best Solution is {0,1,2,4,3,0}
Best Cost is 10
Repeat this process for number of iterations.

0 2 1 4 3 0
Current Cost is 18
0 1 2 4 3 0
Current Cost is 9
0 3 4 2 1 0
Current Cost is 9
0 1 2 3 4 0
Current Cost is 14
0 3 1 2 4 0
Current Cost is 15
0 4 1 3 2 0
Current Cost is 25
0 4 2 3 1 0
Current Cost is 16
0 3 4 2 1 0
Current Cost is 9
0 1 4 3 2 0
Current Cost is 19
0 2 1 4 3 0
Current Cost is 18
0 1 2 4 3 0
Current Cost is 9
0 3 4 2 1 0
Current Cost is 9
0 1 2 3 4 0
Current Cost is 14
0 3 1 2 4 0
Current Cost is 15
0 4 2 3 1 0
Current Cost is 16
0 3 4 2 1 0
Current Cost is 9
0 1 4 3 2 0
Current Cost is 19
0 2 1 4 3 0
Current Cost is 18
0 1 2 4 3 0
Current Cost is 9
Search done!
Best Solution cost found = 9
Best Solution : 0 1 2 4 3 0
BUILD SUCCESSFUL (total time: 1 second)

Some applications of TS

TS has been applied to many combinatorial optimization problems. The aim of this chapter is to describe some of these applications in order to illustrate the ingredients of TS and to give some practical ways of implementing them. More details on these applications and references to other works may be found in: [Hertz & de Werra, 1990]

Conclusions

The above examples have shown how wide the range of applications of Tabu Search is; more descriptions of such cases can be found in [Glover, Taillard, Laguna & de Werra, 1992]. Although basically simple the technique owes its efficiency to a rather fine tuning of an apparently large collection of parameters. In fact experiments have shown that the degrees of freedom in most of these choices are not so restricted. Theoretical considerations based on a probabilistic model of Tabu Search seem to confirm partially this statement [Faigle & Kern, 1992]: when moves from a solution *i* to a solution *j* are performed according to a probability distribution *pij*, then under rather mild assumptions one can prove that convergence is obtained to an optimal solution for probabilities *pij* which are allowed to

be chosen in a large subinterval of [0,1]. This approach may lead to a better understanding of the efficiency of Tabu Search. For the moment, we may just recognize that this technique looks more like an engineering approach to difficult and large size problems of optimization than like an elegant simple mathematical algorithm. With tabu search, complexity is not only present in the problems but in the technique itself. It
is clearly not an advantage, but this situation may be viewed as an exploratory step in a new field of techniques which will likely become more elegant and hopefully more simple when memory, artificial intelligence and exploration procedures will be better integrated in a technique which will probably no longer bear the strange name of tabu.

REFERENCES

[1] H. S. Abdinnour and S. W. Hadley, "Tabu search based heuristics for multi-floor facility layout",
International Journal of Production Research, vol. 38, pp. 365-83, 2000.
[2] M. A. Abido, "A novel approach to conventional power system stabilizer design using tabu search",
International Journal of Electrical Power & Energy Systems, vol. 21, pp. 443-54, 1999.
[3] V. R. Alvarez, E. Crespo, and J. M. Tamarit, "Assigning students to course sections using tabu
search", *Annals of Operations Research*, vol. 96, pp. 1-16, 2000.
[4] F. Glover, J. P. Kelly, and M. Laguna, "Genetic algorithms and tabu search: hybrids for
optimization", *Computers & Operations Research*, vol. 22, pp. 111-34, 1995.
[5] F. Glover and M. Laguma, "Tabu search", in *Handbook of Combinatorial Optimization*, vol. 3, D.
Du and P. M. Pardalos, Eds. Dordrecht: Kluwer Academic Publishers, 1999, pp. 621-757.
[6] F. Glover, M. Laguma, and R. Marti, "Fundamentals of scatter search and path relinking", *Control
and Cybernetics*, vol. 29, pp. 653-84, 2000.
[7] K. Hatta, S. Wakabayashi, and T. Koide, "Adaptation of genetic operators and parameters of a
genetic algorithm based on the elite degree of an individual", *Systems and Computers in Japan*, vol.
32, pp. 29-37, 2001.
[8] M. Maheswaran, S. Ali, H. J. Siegel, D. A. Hensgen, and R. F. Freund, "Dynamic Matching and
Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems", in
Proceedings 8th Heterogeneous Computing Workshop, 1999.
[9] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution programs*. Berlin: Springer-
Verlag, 1994.
[10] M. Mitchell, *An introduction to Genetic Algorithms*. Cambridge, Massachusetts: MIT Press, 1996.
[11] G. Parks, J. Li, M. Balazs, and I. Miller, "An empirical investigation of elitism in multiobjective
genetic algorithms", *Foundations of Computing and Decision Sciences*, vol. 26, pp. 51-74, 2001.
[12] S. Salleh and A. Y. Zomaya, *Scheduling In Parallel Computing Systems: Fuzzy and Annealing
Techniques*. USA: Kluwer Academic Publishers, 1999.
[13]Grzegorz Waligóra, "Simulated Annealing and Tabu Search for Discrete-Continuous Project Scheduling with Discounted Cash Flows", RAIRO - Operations Research / Volume 48 / Issue 01 / January 2014, pp 1-24.