Control and Cybernetics
vol. **29** (2000) No. 3

Tabu Search — a guided tour

by

Magnus Hindsberger and René Victor Valqui Vidal

Department of Mathematical Modelling (IMM)
Technical University of Denmark
DK-2800 Kgs. Lyngby, Denmark
E-mail: mh@imm.dtu.dk ; vvv@imm.dtu.dk

Abstract: The main purpose of this paper is to provide an overview of the ideas behind Tabu Search — one of the most popular metaheuristic appoaches. For the sake of concreteness a simple example of the traveling salesman problem will be used in the discussion to illustrate the process of designing a Tabu Search algorithm. In addition, some extensions will be presented. Finally, applications will also be provided as well as references to more specialized publications.

Keywords: Tabu Search, local search, combinatorial optimization, metaheuristics.

1. Introduction

Tabu Search (TS) was one of the first metaheuristic techniques — for an overview of the methods see Osman and Kelly (1996) — developed to solve combinatorial optimization problems. These problems can be formulated as follows:

Consider a finite *configuration space* (space of configurations or solution space) $S = \{x \mid x = (x_1, x_2, \ldots, x_m)\}$, where m is called the dimension of the space, and a *cost function* $C(x) : S \rightarrow \mathbb{R}$ which assigns a real number to each configuration, to be specific in a minimization problem, we want to find a configuration $x^* \in S$, so that $\forall x \in S$, $C(x^*) \leq C(x)$. Maximization problems are treated analogously.

Interest in metaheuristics is intense because few important combinatorial optimization problems can be solved exactly in a reasonable computer time. Most of these problems arising in practice are *NP-complete*: all known techniques for obtaining an exact solution require an exponentially increasing number of steps as the problem becomes larger. Therefore, emphasis has been directed toward metaheuristic techniques, like TS, for solving these problems.

Graphically, a simple instance of this problem can be visualized as trying to find the lowest point in a complex and highly contoured landscape. An optimization algorithm can thus be viewed as an explorer wandering through valleys and across hilltops searching for the lowest point. The location where the explorer is, can be likened to the current solution x, and the height of this location — to the value of the evaluation function, $C(x)$. Regarded from the point of view of the purpose of the analogy, the neighbourhood $N(x)$ can be defined as the locations found by walking one meter to either the North, South, East, or West.

Staying in the explorer analogy, two LS strategies will be introduced below:

1. the explorer picks randomly *one* of the four neighbourhood locations, evaluates the height, and if the height is lower than the one in his current position, he walks there and continues from this new location picking one of the new neighbours randomly. Otherwise he remains at his old location choosing a neighbour from the neighbourhood again. If he has not changed location for a number of iterations, he is possibly trapped in one of the local extremes (which might be the global one), and the procedure is stopped. This kind of strategy is often called *Greedy*.
2. the explorer evaluates the height of each of the four neighbours. He walks to the best (i.e. lowest) of them and continues from there, as long as it is located lower than the position he came from. Otherwise he is in one of the local extremes, and the procedure is stopped. This strategy, which was also sketched in the previous section, is called *Steepest Descent* (or in case of maximizing, *Steepest Ascent*).

As it can be noted, the explorer in an LS procedure only accepts downhill moves. This severely decreases the probability of finding the global optimal solution as seen in Fig. 2. The explorer is standing at point A, with in this

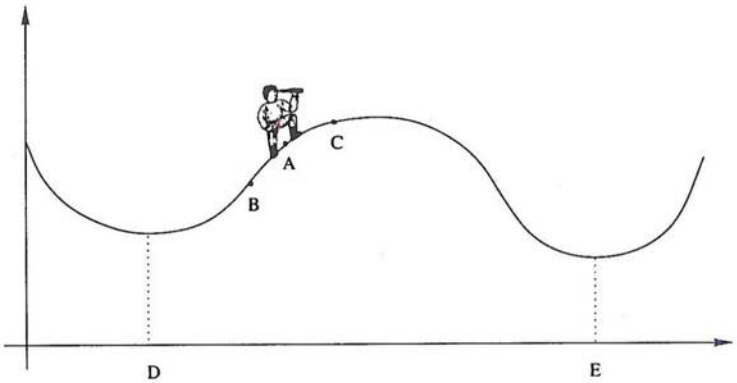


Figure 2. The "landscape" of an unknown function, and an LS operation ex-

example only two neighbours, B and C. He will pick one of them randomly and evaluate its height or he will evaluate both B and C. But he will never choose C as the next current solution, since it is located higher than A. Instead, he will walk downhill toward B and eventually reach the local minimum point D and thus never come to the global minimum in E. Obviously, by rerunning the procedure several times with different start locations, the chance of finding the optimum, or at least a better solution, is improved considerably. Using LS in this way will be described later.

Being fast, the LS strategies are very useful, since they also are both simple and generally applicable, as long as an adequate topology defining the neighbourhood and an evaluation function can be constructed for the problem.

2.2. The traveling salesman problem — part 1

Three things are of importance when implementing local search based methods. The definition of the neighbourhood, $N(x)$, the objective function, $C(x)$, to evaluate the neighbours, and the stopping criterion. How they can be implemented is shown in Example 2.1 below.

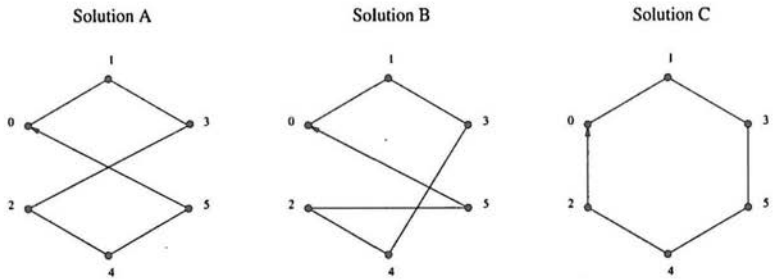
EXAMPLE 2.1 (THE TRAVELING SALESMAN PROBLEM) *The* Traveling Salesman Problem, *TSP, involves the design of a minimum cost path for a salesman, which has to visit a number of cities. Each city must be visited exactly once and the path has to end in the same city it started from.*

In this case the problem has 6 cities numbered 0 through 5, where the salesman has to start from (and thus end in) city no. 0.

A solution can be expressed as:

City no.	0	1	2	3	4	5
Sequence	0	1	3*	2	4*	5

where Sequence *denotes the order in which the cities are visited. For instance, in the solution above, city 2 is visited as number 3 and city 4 as number 4, when the salesman departs from city 0 as illustrated in solution A in Fig. 3.*



THE NEIGHBOURHOOD: *A neighbourhood of the TSP can be defined in several ways:*

- *as moving one city in the sequence to another place in the sequence, e.g. city 1 is visited last instead of first and the other cities are thus visited one "step" earlier.*
- *as an exchange of two cities in the succession they are visited, for instance the two marked with an asterisk *, resulting in the B solution of Fig. 3, where city 2 now is visited 4th and city 4 is visited 3rd. This kind of neighbourhood is traditionally denoted* 2-exchange *in the literature, sometimes with the restriction of not exchanging two cities next to each other in the succession.*

In the following of the example the neighbourhood is the latter of the two mentioned with no restriction.

Note that in this example the column for city 0 must be unchanged, since the city to start from is fixed.

The solution space S is then given by the number of permutations of city 1 through 5, resulting in $5! = 120$ *possible solutions.*

The number of neighbours to each solution is $C(5,2) = 10$. *For a larger number of cities, n, the number of neighbours to each solution would be astronomic, and the Steepest Descent strategy unusable.*

OBJECTIVE COST: *If the cost of traveling between two cities is assumed to be proportional to the distance, the cost of each solution can be calculated from the distance chart below.*

	1	2	3	4	5
0	2.0	2.0	3.5	3.5	4.0
1		3.5	2.0	4.0	3.5
2			4.0	2.0	3.5
3				3.5	2.0
4					2.0

Figure 4. The distance between the cities in the TSP

Assuming the current solution is A, the objective cost, $C(A)$, *of this solution is found as:*

$$C(A) = d(0,1) + d(1,3) + d(3,2) + d(2,4) + d(4,5) + d(5,1) = 16$$

where $d(0,1)$ *denotes the distance between city 0 and city 1. Now, a random*

calculated as:

$$C(B) = d(0,1) + d(1,3) + d(3,4) + d(4,2) + d(2,5) + d(5,1) = 17$$

Since $C(B) > C(A)$ the move to B is rejected and the search from A is continued. Another random neighbour is picked from the neighbourhood, this time C from Fig. 3, by exchanging the number in the succession cities 2 and 5 are visited. By calculating the objective function as before, $C(C)$ is found as:

$$C(C) = d(0,1) + d(1,3) + d(3,5) + d(5,4) + d(4,2) + d(2,1) = 12$$

Now a better solution is found and C can be assigned as the current solution. So in the next iteration the neighbourhood of this point is searched.

STOPPING CRITERION: *It is chosen to continue as above until 20 new neighbours in a row have been rejected, since it is then assumed that a local minimum point has been reached. Other stopping criteria will be discussed in the next sections.*

Whether the Greedy or Steepest Descent strategies should be used depends upon the problem. A Greedy procedure requires only $C(x)$ to be calculated once in each iteration, but has in general more iterations than Steepest Descent, since the latter will go faster to the "closest" minimum point. If the size of the defined neighbourhood is big compared to the computation time of the evaluation function, a Greedy procedure would normally be most efficient. On the other hand Steepest Descent will generally do best with small neighbourhoods. The two strategies can be combined, searching a selected sub-neighbourhood, which in some cases will make the implementation more efficient.

For both methods the computation time of the procedures finding neighbours and calculating $C(x)$ should be optimized, since these procedures will be called many times during a single run.

2.3. Tabu Search

TS can be seen as an improved version of the Steepest Descent strategy. TS utilizes flexible memory to remember a number of the previous steps taken (they will be designated as *tabu*) and will choose other steps in order to exploit new parts of the solution space by taking advantage of history. It is important to note that the same step does not have to be between the same two solutions. If solution A from the TSP example back in Fig. 3 is a result of exchanging the number in the succession cities 1 and 2 are visited and the next step is to solution C, where cities 2 and 5 are exchanged in the sequence, then the step where cities 1 and 2 are exchanged will not bring the situation back to the solution prior to A. But by remembering some information about the previous steps, TS can make sure that the procedure does not alternate between the same two solutions, and, if implemented properly, assures that no cycle between the

In Fig. 5 a pseudocode of TS is shown. The notation $N(x, k)$ is used, since the neighbourhood in iteration k is restricted by the tabu solutions at that time.

```
procedure Tabu Search
begin
  choose initial solution, x ∈ S
  x* := x
  C* := C(x)
  k := 0
  choose V ⊆ N(x, k)
  y* := min{C(y) | y ∈ V}
  while not stop do
    x := y*
    if C(x) ≤ C* then
      x* := x
      C* := C(x*)
    endif
    k := k + 1
    update N(x, k)
    choose V ⊆ N(x, k)
    y* := min{C(y) | y ∈ V}
  end
end
```

Figure 5. Pseudo-code of a Tabu Search procedure

A TS procedure is often stochastic, i.e. it will return different solutions to the same problem. That is so if not the entire neighbourhood of each solution is searched, corresponding to the situation $V = N(x, k)$ in the Steepest Descent strategy, where the best possible solution is chosen, in which case the procedure is deterministic, given the same initial solution is used each time. But, as mentioned in the TSP example, the number of neighbours can be so huge that calculating the cost of all of them will be impracticable. Instead, a smaller part of the entire neighbourhood should be searched. Which part of the neighbourhood is to be searched can either be chosen using some specific rules or random selection — making the algorithm deterministic or stochastic, respectively.

2.4. The traveling salesman problem — part 2

It is quite easy to implement TS, if you already have implemented a Steepest Descent method. The procedure for calculating $C(x)$ is unchanged and the one

The new, and the hardest part, is the implementation of memory. An implementation of TS always includes a short-term memory of the steps taken most recently. This, also called *recency memory*, is used for an *intensification* strategy searching for local minima, while the procedure is still able to overcome those. In addition, a longer term memory can be implemented, remembering the number of times each step has been taken. This is usually referred to as *frequency memory* and is used as a *diversification* strategy to prevent the same steps from being chosen over and over again, as shown in the example below.

EXAMPLE 2.2 (THE TRAVELING SALESMAN PROBLEM — CONTINUED) *How TS, and especially the memory procedure can be implemented is best seen by returning to the previous TSP example. The solution space was the set of permutations of 5 numbers, and a neighbour to a solution was the exchange of two cities in the succession the cities were visited in. A common implementation of memory when using a neighbourhood definition as the one from Example 2.1 is by using an $n \times n$ matrix. In this example $n = 5$ for reasons that should become clear later on.*

Iteration 10

Memory:

Recency

	1	2	3	4	5
1		3			
2	1				2
3				1	
4	2	1	3		
5		2			

Frequency

Current solution:

City	1	2	3	4	5
Seq.	1	3	2	4	5

Objective value: 16

Best Objective value: 15

Neighbourhood:

Swap	Obj. Value
2 - 5	12
2 - 4	17
3 - 5	18

T

Iteration 11

Memory:

Recency

	1	2	3	4	5
1		2			
2	1				3
3					
4	2	1	3		
5		3			

Frequency

Current solution:

City	1	2	3	4	5
Seq.	1	5	2	4	3

Objective value: 12

Best Objective value: 12

Neighbourhood:

Swap	Obj. Value
1 - 3	15
2 - 4	15
3 - 4	16

In Fig. 6 the values assigned to some of the data structures of the TS algorithm during two iterations are shown. The values will be explained below.

The first part is the memory matrix. Its upper triangular part is the recency memory. For iteration 10 it can be seen that the steps, which are tabu during this iteration, are: Exchanging cities 1 and 2 is tabu for the next 3 iterations, exchanging 2 and 5 is tabu for the next 2 iterations, etc. The frequency memory is stored in the lower triangular matrix. For iteration 10 it can be seen that cities 1 and 2 have been exchanged once, cities 1 and 4 twice, etc. for the total of 9 exchanges (since the current iteration number is 10).

The current solution, corresponding to solution A of Fig. 3, is stored as described in Example 2.1, with city 0 left out, since it cannot be exchanged with the others. The objective function value of this solution is 16, but as it can be noted, a better solution, with a value of 15, has been found earlier (this solution is stored too, but is not pictured here).

A neighbourhood of three neighbours has been generated. Neighbour 1 is exchanging cities 2 and 5 resulting in an objective function value of 12. The T *denotes that a given step is tabu, as it can be seen in the recency memory. The other two neighbours are swapping city 2 with 4 and city 3 with 5 resulting in an objective value of 17, respectively 18.*

Which of the neighbours should be chosen? If a step is tabu, it cannot be chosen unless a special condition, called an aspiration criterion, *is satisfied. Some common criteria will be discussed below. The number of iterations a step is tabu, which will be denoted the* tabulength, *can be constant, like above, where a step is tabu for three iterations. Another option is to let the time vary between two bounds, either randomly or systematically. The latter dynamic form will normally prevent the algorithm from cycling between the same solutions.*

The most common aspiration criterion is to accept a tabu step if it results in a better solution than the best found until then. Another case where a tabu step has to be chosen, is when all the steps in the neighbourhood are tabu. Then, the step with the oldest tabu-restriction should be chosen. Other useful criteria can be found in Glover, Taillard, and de Werra (1993), and Glover and Laguna (1993).

Because of the aspiration criterion mentioned it is chosen to swap city 2 and 5 resulting in solution C from Fig. 3. In iteration 11 the data structures have been updated as shown. Note that only two steps are tabu now, since a tabu step was chosen last. But since none of the solutions in the neighbourhood are tabu this time, there will be three tabu steps again from iteration 12.

Frequency memory is usually not as strict as the recency memory. If implemented at all it is normally only used when no solutions are found with a better objective function value than the current best. To the objective function values of the neigbours a penalty, depending upon the frequency of the steps until now, are added. For instance, the penalty in this example could correspond to the

are then 15, 16, and 19, since swapping 3 and 4 has been done three times until now. The neighbour with the smallest adjusted value should be chosen, since it most probably will take the algorithm to the least explored part of the solution space.

The stopping criterion can be to continue for a fixed number of iterations or until no change of x^* or c^* has occurred for a fixed number of iterations.

Because of the memory TS will in general result in very good solutions for most problems. It may be a bit more complicated to implement than Simulated Annealing (Pirlot and Vidal, 1996), but it is still quite easy if procedures for neighbourhood generation and calculation of objective cost either exist or can easily be written. The possible ways to improve the quality of the solutions and/or the computation time will be discussed below.

3. Improvements and extensions

In Glover, Taillard and de Werra (1993) the authors describe several possible improvements of TS. The refinements are divided into three groups: tactical, technical, and computational. A tactical improvement addresses the actual implementation of the problem while technical improvements are general procedures for getting the best results. Computational improvements concern the means to get the best performance on your computer for a specific implementation of the algorithm.

3.1. Tactical improvements

"Good" neighbourhood

Most important, like in any other LS-based heuristics, is that the defined neighbourhood must be "good". When designing your neighbourhood, the use of chunking, that is grouping of basic units of information, as described by Woodruff (1998), might be useful.

In general, one wants the solution space, S, to provide some kind of global convexity of the cost function $C(x)$ (see Hu, Klee and Larman, 1989), which should be also as smooth as possible, i.e. moving to a neighbourhood solution should result in an as small change in $C(x)$ as possible.

In Figs. 7 and 8 we have mapped S for the one dimensional case so that neighbour solutions are assumed to be next to each other when moving along the x-axis. So, the two figures might show the same solution space, but with a different neighbourhood definition. An LS-based heuristic will be much more efficient when using the neighbourhood definition as in Fig. 7, since it would

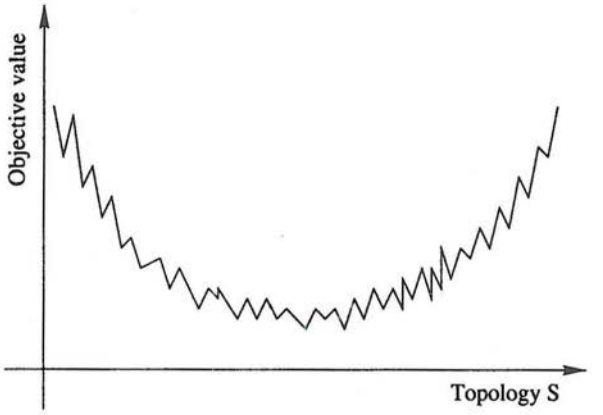


Figure 7. An almost smooth global convex topology of S

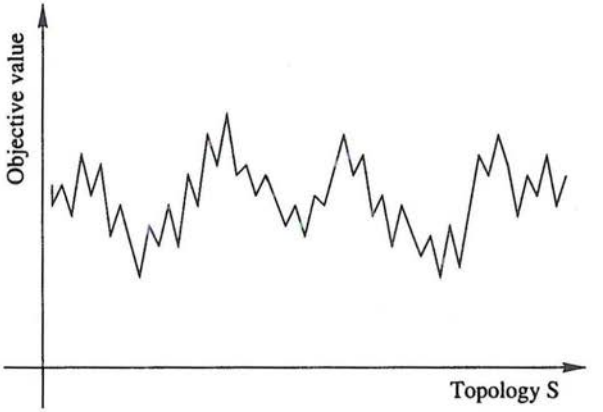


Figure 8. A non-smooth and not global convex topology of S

Strategic oscillation

One tactical technique is the so-called Strategic Oscillation (see Glover, Taillard, and de Werra, 1993; and Glover, Kelly, and Laguna, 1995). In this technique you allow infeasible solutions and add a function to the objective function, which varies between encouraging and discouraging infeasible solutions. This is done by adding a variable penalty for infeasibility to $C(x)$. A sine function is often used for this. Generally it can be sketched as:

where $C'(x)_t$ is the modified objective function value and the function $penalty(t)$, which is the penalty in iteration t, is modified by $infeasibility\_measure(x)$ that describes the degree of infeasibility (if any) of solution x.

There are several reasons for using Strategic Oscillation. Firstly, if the solution space is "non-convex" as in Fig. 8, Strategic Oscillation makes it possible for the algorithm to cross large regions of infeasibility in the search for the optimal solution. It also changes the direction of the search, thus increasing the diversity due to the changing emphasis of the different problem parts given by the definition of the infeasibility measure.

3.2. Technical improvements

The technical improvements do not directly address the problems to be solved. Selecting how many and which neighbours to be evaluated, the tabu list size, etc. are considered technical improvements.

As described in Section 2.2, the number of neighbourhood solutions to be evaluated in each iteration is important. By evaluating all neighbourhood solutions you most usually end up with a high-quality solution, but the quality of the solution may not compensate for the increase in computation time. So only if the time of evaluating a single solution is very short or the number of neighbour solutions is small you will choose to evaluate all neighbours.

The quality of the results of TS is also, like in all other metaheuristics, much dependent upon the chosen parameter values. Analyses like that in Ryan (1995) can help you find good values, but generally the optimal values of the tabu length and the penalty to apply for diversification must be found by parameter analysis.

In general, a highly contoured cost surface, like the one in Fig.8, requires a very long tabu list to overcome the local minima. Clearing all tabu restrictions when a new better value of x^* is found in order to allow unhindered search from this new solution and varying the tabu list size using some transition probabilities are other technical refinements and represent the path of intensification and diversification.

Intensification is the focusing on and exploitation of the promising areas of the solution space. If you find a solution with a better objective function value than before, you know that you have never visited that solution. Therefore, the most efficient search from that point will be using a pure Steepest Descent procedure, which TS performs when no tabu list is defined. So, clearing the tabu list at this point (and afterwards adding elements to the list as usual) will generally improve performance as in Hindsberger and Vidal (2000). Diversification, on the other hand, concerns the exploration of new parts of solution space. The frequency memory helps enforcing the exploration of new areas by penalizing frequently visited solutions or moves made. By using a long tabu list, i.e. the recency memory, the algorithm can escape local minima and thus

efficient exploitation of promising areas by hindering the choice of neighbours for intensification purposes.

Reactive Tabu Search

Reactive Tabu Search (Battiti and Tecchiolli, 1994) is a way of improving both intensification and diversification by making the TS algorithm self-tuning. In Reactive TS you keep track of the number of repetitions (or repeated moves) and increase the size of the tabu list when the percentage of repeated moves grows too high, and decrease it again, when the number of repeated moves has fallen to a lower level. Thus, you get a long tabu list when you need to escape local minima and a small one when you want to close fast on other minima using "unhindered" local search. Fig. 9 shows an example of how the tabu list size can be adjusted as the number of repetitions grows or falls.

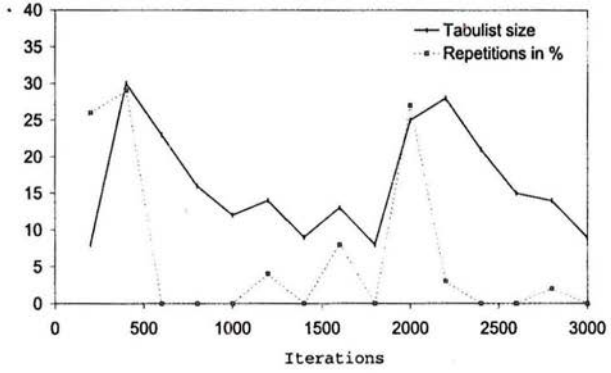


Figure 9. The tabulist size vs. the number of repetitions

Aspiration criteria

The concept of aspiration criteria was described in Example 2.2. In general, if more aspiration criteria are implemented, rules of choice should be defined in order to prioritize them, in case more than one criterion is met simultaneously. Some criteria should only be valid in certain cases, e.g. if the percentage of new solutions visited is too small. More on aspiration criteria can be found in Glover, Taillard, and de Werra (1993), and Glover and Laguna (1993).

3.3. Computational improvements

As computational improvement parallelization of the algorithm counts. Simi-

memory matrix implementation of Fig. 6. Instead of storing the number of iterations a move is tabu and decreasing this number at each iteration until zero it would be better to store the number of the iteration when the move is no longer tabu and then have an iteration counter. Refreshing of the matrix is no longer needed since you can test whether the value in the matrix is higher than the current iteration number to see if it is tabu.

The tabu list of TS serves, as noted by Woodruff and Zemel (1993), two purposes. We quote:

- *Avoidance of cycling — In order to escape a local minimum, the search must be prevented from "falling back" to a recently visited solution. Unless randomness is used in move selection, it is easy to see that if a solution can be revisited, the algorithm may cycle infinitely.*
- *Trajectory — By making certain move attributes tabu, an attribute list often prevents the "reversal" of moves. This results in exclusion of many solutions that have not yet been visited. In many instances this is desirable because it forces the search to explore new regions of S but the aspiration criterion precludes the avoidance of any excellent solutions.*

The distinction above is necessary since the optimal length of the tabu list in order to accommodate these two considerations may be quite different. Therefore, the actual length of the tabu list is a tradeoff between the two considerations above and the fact that a short tabu list makes the search faster and more aggressive.

Woodruff and Zemel (1993) suggest to deal with the problem of cycling by storing a very long list of solutions visited (possibly all of them). A return to an already visited solution is then regarded as tabu, thereby preventing any cycling. In order to store the solutions efficiently you will normally have to code the solutions using a non-bijective hashing function as described in the reference.

3.4. Extensions of TS

Combined approaches

For several applications hybrid approaches containing elements of both the neighbourhood based and the recombining based approaches have proved more successful than using either of them alone.

A pseudocode of the recombining based algorithms like Genetic Algorithms (Goldberg, 1989) looks like this:

1. Construct a set of trial solutions (parent generation)
2. Select the best (from objective function value, diversification aspects, etc.)
3. Mate and combine pairs of solutions into new ones (offspring generation)
4. Modify randomly
5. Make offspring generation the new parent generation and go to 2 until

The main difference is that the recombining based algorithms work with a set of solutions and combine those into new solutions while the local search methods only work with a single solution trying to improve it using small steps.

First attempts of making hybrids were the applications of the LS improvement procedures to Genetic Algorithms, with newly created solutions improved using such techniques. By doing this, the intensification aspect of Genetic Algorithms was improved considerably. This was in effect an advanced form of a multistart LS algorithm (see Glover, Kelly, and Laguna, 1995).

A simple hybrid could look like:

1. Construct a set of trial solutions (parent generation)
2. Apply Local Search improvement heuristic on each solution
3. Select the best of the improved solutions (from objective function value, diversification aspects, etc.)
4. Mate and combine pairs of solutions into new ones (offspring generation)
5. Modify randomly
6. Make offspring generation the new parent generation and go to 2 until stopping criterion is met.

Laporte, Potvin, and Quilleret (1996) describe such an implementation for the Clustered Traveling Salesman Problem, where TS was used as the LS heuristic.

Path Relinking and Scatter Search

Path Relinking and Scatter Search are both intimately related to Tabu Search. Glover, Laguna, and Martí (2000) present in a paper in this special issue the fundamentals of these approaches.

4. Applications

A rather complete list of references related to applications of TS up to 1996 can be found in Soriano and Gendreau (1997) and Glover and Laguna (1997). Therefore, we will complete this paper by listing recent publications related primarily to applications from 1997 till today. This is done in Table 1.

In addition, there is a new application area for TS that seems promising. This is the global optimization of continuous functions. As far as we know the paper by Hu (1992) was the first one dedicated to the adaptation of TS to continuous optimization, though such a reference is not presented in the book by Glover and Laguna (1997). In his paper Hu shows that TS with random moves outperforms the Random Search method and Genetic Algorithms when applied to minimum weight design problems of a three-bar truss, coil springs, a Z-section, and a channel section. For this last case TS with random moves saved 26.14% over other methods.

A recent paper by Siarry and Berthiau (1997) criticizes the paper of Hu because the algorithm proposed is too far from the original TS. Therefore, they

Subject	Reference
Assignment	Scholl and Voss (1997) Hao, Dorne, and Galinier (1998) Chiang (1998) Boumerdassi and Beylot (1999)
Bin packing	Lodi, Martello, and Vigo (1999)
Digital filter design	Fanni, Marchesi, Pilo, and Serri (1998)
Fault section estimation	Fushuan and Chang (1997)
Fuzzy controller	Denna, Mauri, and Zanaboni (1999)
Location	Adenso-Diaz and Rodriguez (1997) Kincaid, Laba, and Padula (1997) Gendron, Potvin, and Soriano (1999)
Molecular recognition	Westhead, Clark, and Murray (1997) Murray, Baxter, and Frenkel (1999)
Pattern classification	Fraughnaugh, Ryan, Zullo, and Cox (1998) Fink and Voss (1999)
Production planning	Hindi (1997) Logendran and Puvanunt (1997) Fink and Voss (1998) Tucci and Rinaldi (1999)
Query optimization	Ribeiro, Ribeiro, and Lanzelotte (1997)
Scheduling	Mazzola and Schantz (1997) Dhodhi and Ahmad (1997) Lopez, Carter, and Gendreau (1998) Colorni, Dorigo, and Maniezzo (1998) Dodin, Elimam, and Rolland (1998) Higgins (1998) McMullen (1998) Macchiaroli, Mole, and Riemma (1999) Zhu and Padman (1999)
Structural design	Bland (1998)
Transmission planning	Zhou, Wang, Ding, Yan, and Li (1999)
Unit commitment	Mantawy, Abdel-Magid, and Selim (1998)
Vehicle routing	Badeau, Guertin, Gendreau, Potvin, and Taillard (1997) Augerat, Belenguer, Benavent, Corberan, and Naddef (1998) Gendreau, Guertin, Potvin, and Taillard (1999)

Table 1

easily be implemented. Tests are performed on classical functions for which minima are known. The examples solved in the two above mentioned papers are of restricted dimensionality, so that more work needs to be done to show the suitability of TS to global optimization as it is the case of Simulated Annealing,

Another new application area that deserves more research is the development of procedures to solve multi-objective combinatorial optimization problems. Several articles in this special issue are dealing with this subject.

5. Conclusions

In this paper we have shown the main decisions to be taken when designing a TS metaheuristic to cope with combinatorial optimization problems. In spite of its simplicity, a look at the available literature documents that TS is one of the most popular metaheuristic approaches that has been applied to a wide spectrum of problems varying from design, planning, scheduling to operational problems that can be formulated as combinatorial optimization models. TS is widely usable and it has become a must in any optimization tool-box as is the case of linear programming, dynamic programming, maximum principles, simulated annealing, evolutionary algorithms, etc.

We have identified the field of continuous global optimization as one area where more research and experimental work ought to be done to be able to demonstrate the suitability of TS compared with other approaches. Another area deserving more research is the multi-objective optimization of combinatorial problems.

The design of a successful metaheuristic approach is a rather complex decision problem that demands problem insight, creativity, knowledge of different approaches (TS is one of them), experimentation, strategy and learning, and many times serendipity. The papers by Borges and Vidal (2000) and Hindsberger and Vidal (2000), in this special issue, are two applications where these ideas have been implemented, with satisfying results.

References

ADENSO-DIAZ, B. and RODRIGUEZ, F. (1997) A simple search heuristic for the MCLP: application to the location of ambulance bases in a rural region. *Location Science*, **5**, 1.

AUGERAT, P., BELENGUER, J.M., BENAVENT, E., CORBERAN, A. and NADDEF, D. (1998) Separating capacity constraints in the CVRP using tabu search. *European Journal of Operational Research*, **106**, 2-3.

BADEAU, P., GUERTIN, F., GENDREAU, M., POTVIN, J.-Y. and TAILLARD, E. (1997) A parallel tabu search heuristic for the vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, **5**, 2.

BATTITI, R. and TECCHIOLLI, G. (1994) The Reactive Tabu Search. *ORSA Journal on Computing*, **6**, 2.

BLAND, J.A. (1998) A memory-based technique for optimal structural design.

BORGES, P.C. and VIDAL, R.V.V. (2000) Fixed channel assignment in cellular mobile telecommunication systems. *Control and Cybernetics*, this issue.

BOUMERDASSI, S. and BEYLOT, A.-L. (1999) Adaptive channel allocation for wireless PCN. *Mobile Networks and Applications*, **4**, 2.

CHIANG, W.-C. (1998) The application of a tabu search metaheuristic to the assembly line balancing problem. *Annals of Operations Research — Paperbound Edition*, **77**.

COLORNI, A., DORIGO, M. and MANIEZZO, V. (1998) Metaheuristics for High School Timetabling. *Computational Optimization and Applications*, **9**, 3.

DENNA, M., MAURI, G. and ZANABONI, A.M. (1999) Learning fuzzy rules with tabu search — an application to control. *IEEE Transactions on Fuzzy Systems*, **7**, 3.

DHODHI, M.K. and AHMAD, I. (1997) Task tree scheduling onto linear arrays using tabu search. *IEE Proceedings — Computers and Digital Techniques*, **144**, 5.

DODIN, B., ELIMAM, A.A. and ROLLAND, E. (1998) Tabu search in audit scheduling. *European Journal of Operational Research*, **106**, 2-3.

FANNI, A., MARCHESI, M., PILO, F. and SERRI, A. (1998) Tabu Search metaheuristic for designing digital filters. *COMPEL — The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, **17**, 6.

FINK, A. and VOSS, S. (1998) Generic application of tabu search methods to manufacturing problems. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, **3**.

FINK, A. and VOSS, S. (1999) Applications of modern heuristic search methods to pattern sequencing problems. *Computers & Operations Research*, **26**, 1.

FRAUGHNAUGH, K., RYAN, J., ZULLO, H. and COX, L.A. (1998) Heuristics for efficient classification. *Annals of Operations Research*, **78**.

FUSHUAN, W. and CHANG, C.S. (1997) A tabu search approach to fault section estimation in power systems. *Electric Power Systems Research*, **40**, 1.

GENDREAU, M., GUERTIN, F., POTVIN, J.-Y. and TAILLARD, E. (1999) Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science*, **33**, 4.

GENDRON, B., POTVIN, J.-Y. and SORIANO, P. (1999) Tabu search with exact neighbor evaluation for multicommodity location with balancing requirements. *INFOR Journal*, **37** (0), 3.

GLOVER, F. (1986) Future paths for integer programming and links to artificial intelligence. *Computers and Operational Research*, **5**.

GLOVER, F., KELLY, J.P. and LAGUNA, M. (1995) Genetic algorithms and tabu search: Hybrids for optimization. *Computers and Operations Research*, **22**, 1.

GLOVER, F. and LAGUNA, M. (1993) Tabu Search. In: C.R. Reeves, ed., *Modern*

GLOVER, F. and LAGUNA, M. (1997) *Tabu Search.* Kluwer Academic Publishers, Boston.

GLOVER, F., LAGUNA, M. and MARTÍ, R. (2000) Fundamentals of scatter search and path relinking. *Control and Cybernetics*, this issue.

GLOVER, F., TAILLARD, E. and DE WERRA, D. (1993) A user's guide to tabu search. *Annals of Operations Research*, **41**.

GOLDBERG, D.E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley.

HANSEN, P. (1986) The steepest ascent mildest descent heuristic for combinatorial programming. In: *Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy.*

HAO, J.-K., DORNE, R. and GALINIER, P. (1998) Tabu search for frequency assignment in mobile radio networks. *Journal of Heuristics*, **4**, 1.

HERTZ, A., TAILLARD, É. and DE WERRA, D. (1997) Tabu Search. In: E. Aarts and J.K. Lenstra, eds., *Local search in combinatorial optimization.* Wiley.

HIGGINS, A. (1998) Scheduling of railway track maintenance activities and crews. *Journal of the Operational Research Society*, **49**, 10.

HINDI, K.S. (1997) Tabu search and applications in production planning. Modern Heuristics for Decision Support, UNICOM Seminars.

HINDSBERGER, M. and VIDAL, R.V.V. (2000) Tabu search for target-radar assignment. *Control and Cybernetics*, this issue.

HU, M. (1992) Tabu search method with random moves for globally optimal design. *Methods in Engineering*, **35**.

HU, T.C., KLEE, V. and LARMAN, D. (1989). Optimization of globally convex functions. *SIAM Journal of Control and Optimization*, **27**.

KINCAID, R., LABA, K.E. and PADULA, S.L. (1997) Quelling cabin noise in turboprop aircraft via active control. *Journal of Combinatorial Optimization*, **1**, 3.

LAPORTE, G., POTVIN, J.-Y. and QUILLERET, F. (1996) A tabu search heuristic using genetic diversification for the clustered traveling salesman problem. *Journal of Heuristics*, **2**.

LODI, A., MARTELLO, S. and VIGO, D. (1999) Approximation algorithms for the oriented two-dimensional bin packing problem. *European Journal of Operational Research*, **112**, 1.

LOGENDRAN, R. and PUVANUNT, V. (1997) Duplication of machines and subcontracting of parts in the presence of alternative cell locations. *Computers & Industrial Engineering*, **33**, 1-2.

LOPEZ, L., CARTER, M.W. and GENDREAU, M. (1998) The hot strip mill production scheduling problem: A tabu search approach. *European Journal of Operational Research*, **106**, 2-3.

MAZZOLA, J.B. and SCHANTZ, R.H. Multiple-facility loading under capacity-based economies of scope. *Naval Research Logistics*, **44**, 3.

MACCHIAROLI, R., MOLE, S. and RIEMMA, S. (1999) Modelling and optimization of industrial manufacturing processes subject to no-wait constraints.

MANTAWY, A.H., ABDEL-MAGID, Y.L. and SELIM, S.Z. (1998) Unit commitment by tabu search. *IEE Proceedings: Generation, Transmission and Distribution*, **145**, 1.

MCMULLEN, P.R. (1998) JIT sequencing for mixed-model assembly lines with setups using Tabu Search. *Production Planning and Control*, **9**, 5.

MURRAY, C.W., BAXTER, C.A. and FRENKEL, A.D. (1999) The sensitivity of the results of molecular docking to induced fit effects: Application to thrombin, thermolysin and neuraminidase. *Journal of Computer-Aided Molecular Design*, **13**, 6.

OSMAN, I.H. and KELLY, J.P., eds. (1996) *Metaheuristics: Theory and Applications.* Kluwer Academic Publishers.

PIRLOT, M. (1992) General local search heuristics in combinatorial optimization. *Belgian Journal of Operations Research, Statistics, and Computer Science*, **32** (1,2).

PIRLOT, M. and VIDAL, R.V.V. (1996) Simulated annealing: A tutorial. In: R.V.V. Vidal and Z. Nahorski, eds., Simulated Annealing Applied to Combinatorial Optimization. *Control and Cybernetics*, **25**, 1.

RIBEIRO, C.C., RIBEIRO, C.D. and LANZELOTTE, R.S.G. (1997) Query optimization in distributed relational databases. *Journal of Heuristics*, **3**, 1.

RYAN, J. (1995) The depth and width of local minima in discrete solution spaces. *Discrete Applied Mathematics*, **56**.

SCHOLL, A. and VOSS, S. (1997) Simple assembly line balancing — heuristic approaches. *Journal of Heuristics*, **2**, 3.

SIARRY, P. and BERTHIAU, G. (1997) Fitting of tabu search to optimize functions of continuous variables. *International Journal for Numerical Methods in Engineering*, **40**.

SILVER, E.A., VIDAL, R.V.V. and DE WERRA, D. (1980) A tutorial on heuristic methods. *European Journal of Operational Research*, **5**.

SORIANO, P. and GENDREAU, M. (1997) Fondements et applications des méthodes de recherche avec tabous. *Recherche Opérationelle*, **31**, 2.

TUCCI, M. and RINALDI, R. (1999) From theory to application: tabu search in textile production scheduling. *Production Planning and Control*, **10**, 4.

WESTHEAD, D.R., CLARK, D.E. and MURRAY, C.W. (1997) A comparison of heuristic search algorithms for molecular docking. *Journal of Computer-Aided Molecular Design*, **11**, 3.

WOODRUFF, D.L. and ZEMEL, E. (1993) Hashing vectors for tabu search. *Annals of Operations Research*, **41**.

WOODRUFF, D.L. (1998) Proposals for chunking and tabu search. *European Journal of Operational Research*, **106**.

ZHOU, L., WANG, X., DING, X., YAN, Z. and LI, S. (1999) Application of genetic algorithm/tabu search combination algorithm in distribution network structure planning. *Power System Technology*, **23**, 9.

ZHU, D. and PADMAN, R. (1999) A meta-heuristic scheduling procedure for resource-constrained projects with cash flows. *Naval Research Logistics*,

CALL FOR PAPERS ANNOUNCEMENT

The Fourteenth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems
(IEA/AIE-2001)

Budapest, Hungary, June 4–7, 2001

Sponsored by the International Society of Applied Intelligence and cooperated with major international organizations, including ACM/SIGART, AAAI, INNS, IEE, CSCSI, JSAI, ECRIM, HAS, and SWT.

Submit the required materials to Dr. Laszlo Monostori.

Authors can obtain further details of the call for papers announcement either from the conference web page,

`www.sztaki.hu/conferences/ieaaie2001`

or from Dr. Laszlo Monostori,
Program Chair IEA/AIE-2001,
Computer and Automation Research Inst.,
Hungarian Academy of Sciences,
Kende u. 13-17, H-111 Budapest, Hungary.

Fax +36 1 466 7503
E-mail `ieaaie2001@sztaki.hu`