1

Tabu Search: A Comparative Study

Harun Pirim, Engin Bayraktar and Burak Eksioglu
Mississippi State University, Industrial and System Engineering Department
USA

1. Introduction

Problems encountered in fields like scheduling, assignment, vehicle routing are mostly NP-hard. These problems need efficient solution procedures. If confronted with an NP-hard problem, one may have three ways to go: one chooses to apply an enumerative method that yields an optimum solution, or apply an approximation algorithm that runs in polynomial time, or one resorts to some type of heuristic technique without any a priori guarantee for quality of solution and time of computing (Aarts & Lenstra, 2003). Heuristics fall under the general heading of local search approaches. Hence, local search techniques are widely used to find "close-to-optimum" solutions to these problems in a "reasonable" amount of time. Tabu search (TS) is one of the most efficient heuristic techniques in the sense that it finds quality solutions in relatively short running time. This chapter will provide a basic description of TS giving insights for novice readers as well as introduce application areas and provide comparisons of TS to other meta-heuristic procedures for the readers with more experience on local search procedures.

The chapter will be organized as follows: The second section is going to introduce the basic terminology. For example, definitions for global optimization, local search, heuristics, and meta-heuristics will be provided. The section will also provide brief descriptions of TS as well as the following meta-heuristics to which TS will be compared: simulated annealing (SA), genetic algorithms (GA), ant colony optimization (ACO), greedy randomized adaptive search procedure (GRASP), and particle swarm optimization (PSO). Second section is intended to give the readers a good overall view of the "local search" area and let them know that TS will be compared to several other meta-heuristic procedures.

In the third section, basic steps of TS, SA, GA, ACO, GRASP and PSO will be described. As the mechanisms of these procedures are explained, differences and similarities between TS and each of the other procedures will be pointed out. Section three will familiarize the readers with the various meta-heuristic procedures that will be discussed throughout the chapter.

The fourth section will be dedicated to identifying the different problems for which TS was used to generate solutions. For example; TS has been used to solve scheduling problems, routing problems, and assignment problems. We will try to generate a comprehensive list of the problems to which TS has been applied. This section will provide the reader with an understanding of how TS has been used.

In the fifth section, efficiency and effectiveness of TS will be compared to other meta-heuristic procedures. Reasons why TS is more efficient and/or effective than some of the

Source: Local Search Techniques: Focus on Tabu Search, Book edited by: Wassim Jaziri, ISBN 978-3-902613-34-9, pp. 278, October 2008, I-Tech, Vienna, Austria

other local search techniques will be discussed. Section five will explain why TS is the choice of solution method for some problems and not for others. Section six will conclude the chapter.

2. 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.

2.1. General definitions

The term "combinatorial" refers to the constraint that the solution set has to be finite or countably infinite (Michiels et al., 2007). Many combinatorial optimization problems can be expressed as a search for a specific permutation (Dréo et al. 2006). Solution space of combinatorial optimization problems can typically be represented by sequences, permutations, graphs and partitions (Michiels et al., 2007).

Combinatorial optimization problem: Optimizing a linear function subject to other linear functions over a finite (or countably infinite) set of possible solutions is called a combinatorial problem. Combinatorial optimization is the discipline of decision making in case of discrete alternatives (Aarts & Lenstra, 2003). In other words, in combinatorial optimization, one looks for an object from a finite, or countably infinite set, permutation, or graph (Papadimitriou & Steiglitz, 1998).

Global and local optimum: An optimization problem with a feasible solution set S and a neighborhood function N has a local optimal solution that is also globally optimum if N is exact. A solution is locally optimum if and only if its out degree is zero in the transition graph which is a directed, acyclic sub-graph of a neighborhood graph. A globally optimum solution can be found within a small number of steps if the neighborhood graph is strongly connected, which means for each pair of solutions (a, b), b is reachable from a, and its diameter (maximum distance between any pair of solutions) is not too large. If a graph is not strongly connected then its diameter is infinitely large. A local optimal solution to a problem may be poor (i.e. far from the global optimum). Hence, a better solution can be generated by applying a more powerful neighborhood function which obviously is a trade-off between quality of a solution and computation time to yield that solution.

Complexity: "A measure of computer time or space to solve a problem by an algorithm as a function of the problem's dimensions. Suppose $T(n)$ is the time it takes to solve an instance of a problem with dimension n. Then, the algorithm has (worst-case) time complexity K(n), if the greatest time it could take to solve an instance of the problem is O(K(n)). When K(n) is a polynomial, we say the algorithm has polynomial time complexity"(Holder, 2006). If the running time of an algorithm is not polynomial then it is typically exponential. For example, if we try to find the best tour for a Travelling Salesman Problem (TSP) with one hundred cities, the number of solutions exceeds 10^{50} which is larger than the estimated number of particles in the universe (Michiels et al., 2007). If a problem is polynomially reducible to

another problem then the new problem is at least as hard as the old one and a polynomial-time algorithm exists for the new problem if and only if it exists for the old problem.

Heuristics, meta-heuristics, hyperheuristics: Heuristic usually refers to a procedure that seeks an optimum solution but does not guarantee it will find one, even if one exists. Meta-heuristics are general frameworks for heuristics in solving hard problems. The idea of "meta'' is that of level (Holder, 2006). Meta-heuristics do not stop in the first local optimum as a simple heuristic does. They can be classified into two: those that perform a single walk in the neighborhood graph using special procedures trying not to be trapped in a local optimum and those that perform multiple walks (Michiels et al., 2007). TS and SA are examples for the first class. Hyperheuristics choose between given heuristics at various decision points in an optimization problem.

Constructive algorithm: An algorithm that generates a solution through a number of steps where in each step a partial solution is obtained and a complete solution is obtained in the final step.

Plateaus: A part of a solution space that contains solutions with the same objective function value.

Local search algorithm: An algorithm that searches through the solution space and tries to find good quality solutions in each step by means of a neighborhood.

Graph representation of solutions may be inspiring for the designer to be able to direct the search more intelligently (Dréo et al. 2006). For a local search algorithm to be effective, solution space of the problem should not comprise large plateaus. Plateaus may cause cycling. There are ways of avoiding cycling such as remembering recently visited solutions as in TS short term memory.

Local search is what we always do when we are supposed to find a solution in practical life as well. Local search associates by local optimum and local optimum may be a step/stop for global optimum. One may try to modify a local optimal solution in order to get a better solution. However, it is necessary to prevent cycling among solutions visited. This probability of revisiting a previously visited solution is inevitable unless necessary cautions are taken. In that sense, TS uses memory property to prevent cyclic motions in the solution space. TS uses short-term and/or long-term memory while making moves between neighboring solutions. It is essential for a local search to be balanced in terms of quality of solutions and computing time of these solutions. In that sense, a local search does not necessarily evaluate all neighborhood solutions. Generally, a subset of solutions is evaluated.

We can give a maze analogy to explain how local search works: a man needs to find the door to get out of the maze. All paths he travels look similar. He goes back and forth to find the exit door. He makes moves to be able to search through the maze. He has some signs-such as colorful marbles- which he can put on the floor of areas he walked through in order to understand that he visited these areas previously so that he can narrow his search space and easily find the door which will lead him outside. In this analogy, the maze represents the search space of neighborhood solutions of a problem instance. Moves made by the man are the moves (iterations) of an algorithm. His back and forth moves may represent cycles. Marbles are rules (e.g. tabu list) that prevent an algorithm from being trapped in a cycle or

local optimum and narrow the search space. The door opening outside represents a local or global optimal solution of a problem instance.

A local search algorithm begins with an initial solution. This initial solution can be generated by any heuristic algorithm. The algorithm then searches through the solution space with guidance of a neighborhood function. In other words, the algorithm makes a walk through the neighborhood graph. There are different strategies for walking through a neighborhood graph. The most obvious strategy is used by the iterative improvement algorithm also known as the hill climbing algorithm (Michiels et al., 2007). This basic algorithm searches for a better solution in the neighborhood. If it finds a better solution, it changes the current solution with this new one. Otherwise, the algorithm stops and keeps the current local optimum solution (Figure 2.1). The simplex method of linear programming is also a hill climbing procedure that moves from one extreme point solution to another, using an exact neighborhood.

Algorithm Hill Climbing (Iterative improvement)
begin
i:=initial solution
repeat
generate an *s* ϵ *N(i);*
if f(*s*) > f(*i*) **then** *i*:=*s*;
until f(*s*) ≤ f(*i*) for all s ϵ *N(i);*
end;

Figure 2.1 Hill climbing algorithm

In iterative improvement, selecting a better solution within a neighborhood is done using what's called a pivoting rule. A pivoting rule generally selects the first better solution or the best solution within a neighborhood. We had mentioned that we could represent the solution space as a transition graph. The potential of such a graph gives a lower bound on the number of iterations that are maximally required by iterative improvement. One disadvantage of applying iterative improvement is the possibility of being trapped at a local optimum. In order to escape from such a possibility, the neighborhood function can be defined accordingly. Another alternative is allowing non-improving moves as well as performing multiple runs of iterative improvement (Michiels et al., 2007). For example, TS and SA allow non-improving moves as it will be discussed in more detail in the following sections.

Neighborhood function: Given a feasible solution *s*, we can define a set of solutions *N(s)* elements of which are close to *s*. Neighborhood functions can be generated based on the structure of the problem at hand. Some basic neighborhoods are inversions of two elements placed successively in the permutation, transposition of two distinct elements, or displacement of an element (Dréo et al. 2006). If a solution *s* is neighbor of solution *s'* and *s'* is also a neighbor solution of *s* then *N i*s called a symmetric neighborhood function. Neighborhood functions can be defined as swapping, moving, replacing operations. A neighborhood is said to have a performance bound *C* if all local optimum costs have at most a cost of *C* times optimum cost. If the neighborhood function is exact then iterative improvements end up with an optimal solution.

2.2 Tabu search definitions

Short-term memory: A kind of memory that is limited in terms of time and storage capacity. In TS, the tabu list can be regarded as a short-term memory. Recency memory which will be defined later is also a short-term memory. With a short term memory, a previsited solution may be revisited with a different neighborhood.

Long-term memory: A kind of memory that differs from short-term memory in terms of time and storage capacity. The probability of visiting a previously visited solution using long-term memory is very small. Intensification and/or diversification are/is achieved through long-term memory.

Move: A modification made to a solution. Local search will tend to visit previously visited solutions more frequently if the number of tabu moves are relatively small. On the other hand, if the number of tabu moves is large then the search is less probable to find good local optima due to lack of available moves.

Tabu list: In order to prevent visiting a previously visited solution, TS uses a tabu list in which tabu moves or attributes of moves are listed. Also, tabu name is originated by these prohibited move states. Short tabu lists may not prevent cycling resulting in information loss while long tabu lists may excessively prevent neighborhood so that moves are limited to some extent.

Aspiration: Some tabu moves need to be disregarded in order to obtain better local solutions. These moves are called aspired moves. "Improved-best" and "aspiration by default criteria" are examples. Moving to a solution better than the last solution found so far is a commonly used aspiration criterion as well.

Probabilistic tabu search: Unlike many TS applications which have deterministic moves, probabilistic TS assigns a probality for each move (Glover & Laguna, 1997). Convergence theorems for SA can be adapted to probabilistic tabu conditions.

Quality dimension of memory: Ability to differentiate the merit of solutions to problem instance visited during the search (Glover & Laguna, 1997). Using a quality dimension, memory can be used to describe the elements of a good solution so that a bad move can be penalized.

Recency-based memory: A kind of short-term memory that keeps tracks of solution attributes that have changed during the recent past. This is the most described TS feature in the literature (Glover & Laguna, 1997).

Frequency-based memory: To make sure that a certain level of diversity is maintained throughout the search without prohibiting many moves, frequently used moves can be penalized (Dréo et al. 2006). Variety of penalizing methods can be derived. Frequency of certain moves not exceeding a predefined threshold is an example. It is obvious that a penalty mechanism must be used along with an aspiration criterion not to miss good solutions. Penalty based on frequencies may be regarded as a long-term memory. In general, it is better to use both long-term and short-term memory together. Such collaboration can be either "constant" or "varied". The term "constant" indicates that the number of tabu moves and penalty coefficients are predetermined. On the other hand, "varied" refers to alternating search phases. The purpose of these phases will be to intensify the search or to diversify the search. Intensification is related to reducing the number of prohibited moves and/or evoking the long-term memory to choose solutions with characteristics close to the best solutions enumerated by the search. Diversification is achieved to some extent by means of a short term memory.

Intensification aims to guide the search through a part of the solution space that is likely to have promising solutions while diversification aims to force the search through unvisited areas in the solution space.

Critical event memory: A kind of memory that monitors the occurrence of critical events and constitutes an aggregate summary of these events (Glover & Laguna, 1997).

Explicit memory: Records complete solutions generally consisting of elite solutions (Glover & Laguna, 1997). This kind of memory needs excessive space, and it expands neighborhood during the search. Explicit memory is related to intensification.

Attributive memory: Records information about solution attributes rather than about solution. In the TSP, index of tours may be used as attributes. Attributive and explicit memories complement each other. Attributive memory reduces the neighborhood size by forbidding certain moves.

Tabu-active: Attributes that change as a result of a move. These attributes may be used in a recency-based memory called "tabu-active" for a specified number of iterations.

Tabu tenure: Size of a tabu list or previous solutions to be stored. Effective tabu tenures depend on the size of a problem instance.

Strategic Oscillation: A means for interplay between intensification and diversification over a long term (Figure 2.2)

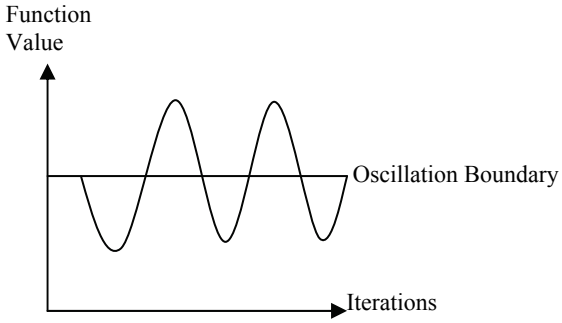

Figure 2.2 Graph of Oscillation Boundary

Tabu Thresholding: A method that joins prescriptions of strategic oscillation with candidate list strategies. Thresholding methods guide greedily exploring search space in a nonmonotonic way.

Hash functions: These are used in TS to provide a mechanism to avoid cycling in such a way that it won't result in expensive computation. Solution vectors can be mapped to integers that can be stored for a number of recent iterations.

Path relinking: A kind of integration of intensification and diversification strategies. Based on this approach, new solutions are generated exploring trajectories that connect elite solutions by generating a path in the neighborhood space (Glover & Laguna, 1997).

Vocabulary building: It is about generating new attributes out of other attributes like generating new populations or offsprings in GA. The basic idea is identifying meaningful parts of solutions as a basis to generate new combination of solutions. It can be viewed as an instance of path relinking.

2.3 Other meta-heuristic definitions

All meta-heuristics, including TS, have terminology and stories associated with them. For example, some meta-heuristics are inspired from animal behaviors, others from biology, and some from manufacturing processes. In this part of the chapter, we will concentrate on the definitions related to meta-heuristic procedures other than TS.

2.3.1 Simulated annealing (SA)

The SA procedure is inspired from the annealing process of solids. SA is based on a physical process in metallurgy discipline or solid matter physics. Annealing is the process of obtaining low energy states of a solid in heat treatment. Annealing process starts with melting the solid by heat treatment. Particles constituting the solid are arranged according to that heat treatment. Then, temperature is decreased which results in minimum energy state.

Threshold Value: Value from which difference of costs associated with two solutions should be strictly less.

Temperature (control parameter): It is the expected value of the thereshold.

2.3.2 Genetic algorithms

Chromosomes: These are strings of parameters which construct proposed solutions to the problem.

Crossover: In order to find better solutions and maintain diversity, crossover is used in genetic algorithms. For instance, a new solution (a child) can be obtained by combining two separate solutions (parents). This combination is defined as crossover. Figure 2.3 provides an example. There are two chess boards and pawns on these boards. We divide the chess boards from the middle into two parts. On the first chess board, we have 3 pawns on the left and 3 pawns on the right. On the second chess board, we have 6 pawns on the left and 4 pawns on the right. An example of crossover would be combining the left half of the first board with the right half of the second board. It is possible to obtain diverse solutions in genetic algorithms by utilizing the crossover operation. However, crossover may lead to infeasible solutions, and one needs to be aware of such situations.

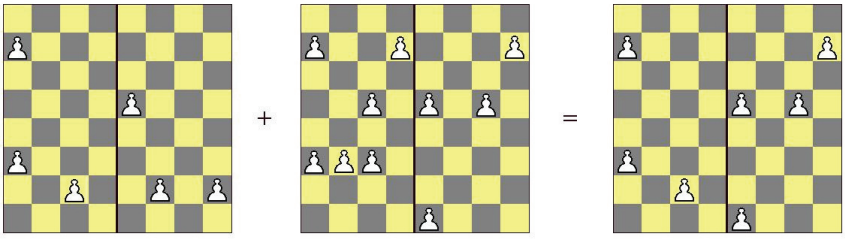

Figure 2.3 A crossover example considering different pawn orders on the chess board

Evolutionary Algorithms: Algorithms which are very similar to genetic algorithms. They are population based algorithms and they rely on artificial intelligence.

Fitness Function: A tool used in genetic algorithms that serves as a neighborhood function so that different solutions can be compared based on the values obtained from the fitness function.

Initialization: A step in which initial solutions are obtained by using initial populations. These initial populations are usually generated randomly. Using these populations, possible solutions are determined some of which are selected to construct the search space.
Mutation: An operator which is used to provide diversity in between populations.
Selection: This is the process in which the solutions are selected in accordance with the results from the fitness function.
Reproduction: After obtaining the results through crossover and/or mutation, another population is generated. From this newly created population, again new children are determined by the help of crossover and/or mutation and the reproduction process goes on.
Termination criterion: A condition that indicates when the algorithm will stop.

2.3.3 Ant colony optimization

Ants: Small animals (insects) that live in colonies in/on the ground. With this real life definition, ant colony optimization is an optimization method in which imaginery agents are used.
Daemon Actions: These are the actions that can be taken to centralize the solution. The aim of Daemon Actions is to prevent quick convergence of the algorithm.
Decentralized Control: A term which is related to robustness and flexibility. Robust systems are desired because of their ability to continue to function in the event of breakdown of one of their components (Dréo et al., 2006).
Dense Heterarchy: A term which is taken from biology and represents the organization of ant colonies. It is different from the managerial term hierarchy. In dense heterarchy, the structure is horizontal, contrary to hierarchy (see Figure 2.4).
Pheromone: In real life, pheromone refers to the chemical material that an ant spreads over the path it goes and the level of it changes over time by evaporating. On the other hand, in ant colony optimization, pheromone is a parameter. The amount of this parameter determines the intensity of the trail. The intensity of the trail can be viewed as a global memory of the system (Dréo et al., 2006).

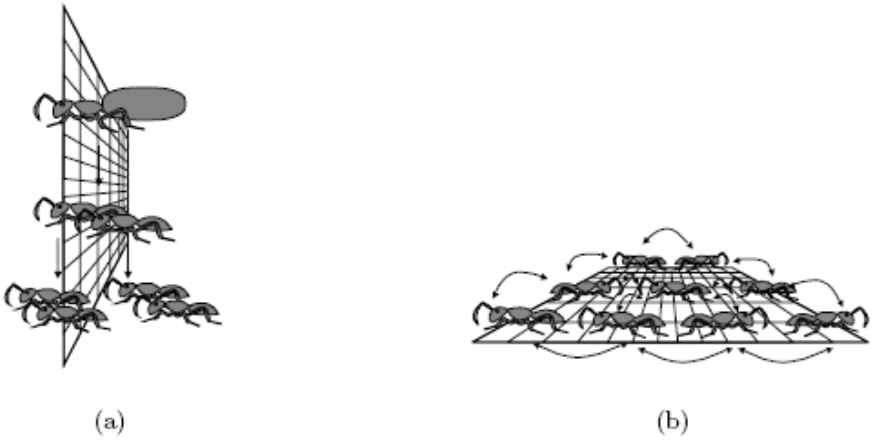

Figure 2.4 - Hierarchy (a) and dense heterarchy (b): two opposite concepts (Dréo et al., 2006)

MAX-MIN Ant System: An improved version of the "Ant System" in which only the best ant updates a trail of pheromone and values of the trails are limited, the maximum value is given to the trail initially (Dréo et al., 2006).
Positive Feedback: Feedback that instructs all ants to follow the same single path to reach the solution.
Stigmergy: The indirect communication among ants when finding a path to reach the food.

2.3.4 GRASP

Greedy Function: The function is used to rank the solution elements.
Iterated Local Search: It is the search in which a locally optimal solution is derived by using iterative improvement of GRASP.
Restricted Candidate List (RCL): A list in which well ranked elements of partial solutions are placed.

2.3.5 Particle swarm optimization

Cognitive Consistency: A meaningful pattern among the particles in a given society. Particles can be anything from animals to cities depending on the system studied. Based on cognitive consistency, when one group of particles thinks in a certain way, the other groups also think in the same or a similar way.
Global Best: The best position found after an update made by any particle in the swarm.
Local Best: The best solution that a particle has seen.
Neighborhood Best: It is the best solution that the particle finds after examining the neighboring solutions.
School: The set of elements (e.g. animals) in a society.
Position: The location of the particle in a specific iteration.
Social Influence: A term used in PSO that describes the logic of particle swarm. In real life, people have thoughts and these thoughts can change after social interactions like conversations. The same logic is used for PSO so that the solutions are changed in the best possible way.
Swarm Intelligence: The artificial intelligence that is made up of simple agents that interact with one another and with the environment.
Velocity: The direction of movement of the particles of a particular society.

3. Meta-heuristic procedures

It is possible to classify meta-heuristics in many ways. Different view points differentiate the classifications. Blum and Roli (2003) classified meta-heuristics based on their diverse aspects: nature-inspired (e.g. GA, ACO) vs. non-nature inspired (e.g. TS); population-based (e.g. GA) vs. single point search (also called trajectory methods, e.g. TS); dynamic (i.e. guided local search) vs. static objective function; one vs. various neighborhood functions (i.e. variable neighborhood search); memory usage vs. memory-less methods. A classification of meta-heuristics is given in the Table 3.1 in which "A" represents the adaptive memory property, "M" represents the memory-less property, "N" represents employing a special neighborhood, "S" represents random sampling, "1" represents

iterating-based approach, and "P" represents a population-based approach. Population-based approaches, also referred to as evolutionary methods, manipulate a set of solutions rather than one solution at a stage.

Meta-heuristic	Classification
Tabu-search	A/N/1-P
Simulated annealing	M/S-N/1
GA	M/S-N/P
ACO	M/S-N/P
GRASP	M/S-N/1
PSO	M/S-N/P

Table 3.1 – Classification of Meta-heuristics (modified from Glover, 1997)

Almost all meta-heuristic procedures require a representation of solutions, a cost function, a neighborhood function, an efficient method of exploring a neighborhood, all of which can be obtained easily for most problems (Aarts & Lenstra, 2003). It is important to mention that a successful implementation of a meta-heuristic procedure depends on how well it is modified for the problem instance at hand.

3.1 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. These restrictions are in relation to memory structures that can be thought of as intelligent qualifications. Intelligence needs adaptive memory and responsive exploration (Glover & Laguna, 1997). For example, while climbing a mountain one remembers (adaptive memory) attributes of paths s/he has traveled and makes strategic choices (responsive exploration) on the way to peak or descent. TS also uses responsive exploration because a bad strategic decision may give more information than a good random one to come up with quality solutions. 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 3.1.

```
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 3.1 – A basic tabu search algorithm

where T is a tabu list and *N(s)* is the set of neighborhood solutions. A generic flowchart of TS algorithm can be given as follows in Figure 3.2:

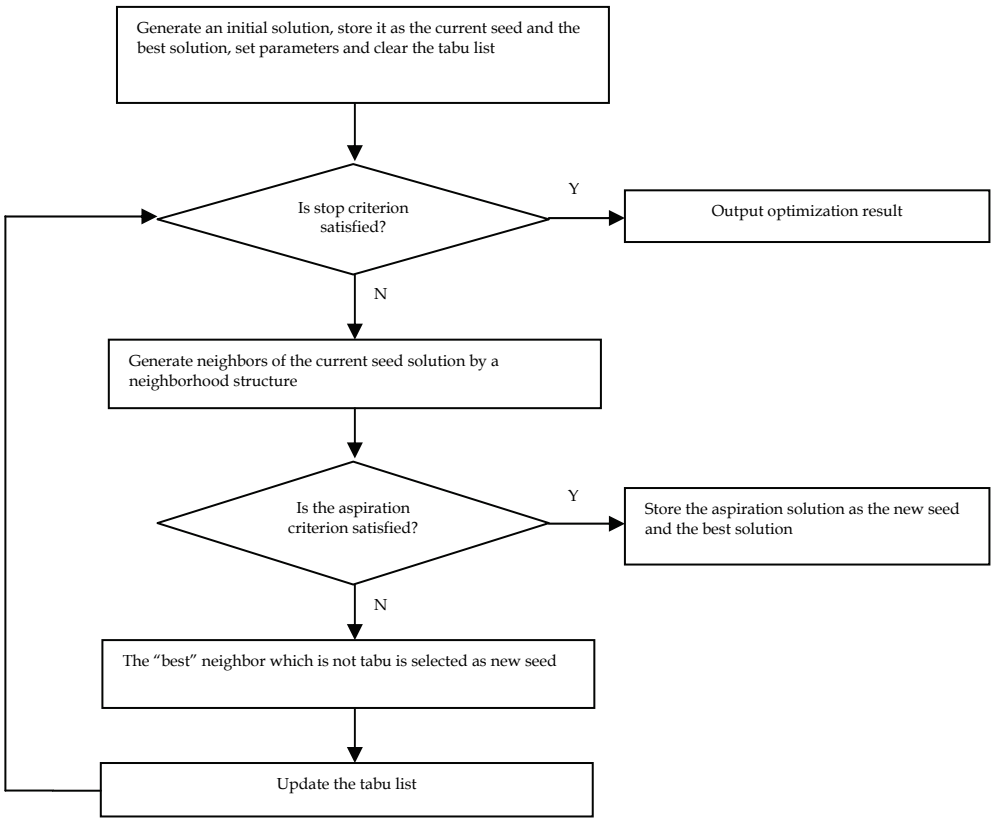

Figure 3.2 - Generic flowchart of TS algorithm (Zhang et al. 2007)

TS memory can be implemented by means of matrices as shown in the practical example below provided in Figure 3.3 (Hindsberger & Vidal, 2000) for the TSP:

Iteration 10

Recency M.

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

Frequency M.

Current solution:

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

Objective Value: 16

Best Objective Value: 15

Neighborhood:

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

Iteration 11

Recency M.

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

Frequency M.

Current solution:

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

Objective Value: 12

Best Objective Value: 12

Neighborhood:

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

Figure 3.3 – Matrix implementation of recency memory (Hindsberger & Vidal, 2000)

Here, upper triangular matrix represents recency memory which stores tabu moves. For example, in iteration 10 exchanging cities 1 and 2 is tabu for 3 iterations, exchanging cities 2 and 5 is tabu for 2 iterations etc. Lower triangular matrix represents frequency memory which stores frequency of exchanging cities. For example, in iteration 10 cities 1 and 2 were changed once etc. Total number of exchanges is 9 since it is the 10^{th} iteration we are in. T represents the tabu move.

3.2 Simulated Annealing (SA)

SA is a randomized algorithm that tries to avoid being trapped in local optimum solution by assigning probabilities to deteriorating moves. In SA a threshold value is chosen. The increase in cost of two moves is compared with that threshold value. If the difference is less than the threshold value, then the new solution is chosen. A high threshold value may be chosen to explore various parts of solution space while a low threshold value may be chosen to guide the search towards good solution values. The threshold value is redefined in each iteration to enable both diversification and intensification. Starting with high threshold

values and then decreasing the value may result in finding good solutions. SA uses threshold as a random variable. In other words SA uses expected value of threshold. In a maximization problem acceptance probability of a solution is defined as follows:

$$IP\{s'\} = \begin{cases} 1 & f(s') \geq f(s) \\ \exp\left(\dfrac{f(s') - f(s)}{c_k}\right) & f(s') < f(s) \end{cases}$$

where c_k is the temperature that gives the expected value of the threshold. A generic SA algorithm for a maximization problem is given in Figure 3.4 below:

```
algorithm Simulated annealing
begin
    s:= initial solution
    k:=1;
    repeat
        generate an s′ ϵ N(s);
        if f(s′) ≥ f(s) then s:=s′
        else
            if exp( (f(s') − f(s)) / c_k ) > random[0,1) then s:=s′;
        k:=k+1;
    until stopcriterion:
end;
```

Figure 3.4 – A simulated annealing algorithm

The cooling schedule is important in SA. Temperature values (c_k) are specified according to the cooling schedule. In general, the cooling schedule's temperature is kept constant for a number of iterations before it is decreased.

3.3 Genetic Algorithms (GAs)

GAs are used to create new generation of solutions among trial solutions in a population. GA is a population-based heuristic that imitates a biological system to find a reasonable solution to a difficult problem. In this section, we will observe how one can obtain efficient solutions with respect to computation time and solution quality using GAs.

In a GA, a "fitness function" is utilized and hence a quantitative study is performed. The fitness function evaluates candidate solutions, determines their weaknesses and deletes them if they are not expected ones. After this step, the reproduction among the candidates occurs and new solutions are obtained and compared using the fitness function again. The same process keeps repeating for number of generations.

With the above description in mind, Figure 3.5 shows a general schema of using GA for minimization problems. The initial step is to determine P_0, the first population of solutions. Using the fitness function, improvements are made to the initial population of solutions. Afterwards, the algorithm enters into a loop in which crossover and mutation operations are performed until a stopping criterion is met. A typical stopping criterion is to perform all the steps for a fixed number of generations.

```
Begin
P0 := set of N solutions;
/*Mutation*/
replace each s ∈ P0 by Iterative_Improvement(s);
t :=1;
repeat
Select Pt ⊆ Pt-1;
/* Recombination */
extend Pt by adding offspring;
/* Mutation */
replace each s ∈ Pt by Iterative_Improvement(s) ;
t :=t+1;
until stopcriterion;
end;
```

Figure 3.5 - A genetic local search algorithm for a minimization problem (Michiels et.al., 2003)

GAs have many application areas in Aerospace Engineering, Systems Engineering, Materials Engineering, Routing, Scheduling, Robotics, Biology, Chemistry, etc.

3.4 Ant Colony Optimization (ACO)

ACO is another branch of meta-heuristics that is used to solve complex problems in a reasonable amount of time.
In Figure 3.6, a general type of ant colony optimization is given.

```
procedure ACO_Meta-heuristic
        while (not_termination)
        generateSolutions()
        pheromoneUpdate()
        daemonActions()
        end while
end procedure
```

Figure 3.6 - A general ant colony optimization procedure

As seen from the general algorithm, a set of initial solutions should be generated in each turn of the while loop, then the pheromone levels should be updated and actions should be taken. When the termination criterion is reached, the procedure ends. This algorithm can be modified to fit the needs of the specific problem.
In Figure 3.7, a generic ACO procedure is provided to solve the Traveling Salesman Problem (TSP). J_i^k is the list of already visited cities.

As shown in Figure 3.7, the algorithm begins with generating an initial solution. The variable m represents the number of ants. Each city is selected based on a probability function which is given in *(1)* and after the selection; the evaporation is performed by utilizing equation *(2)*. However, in order to calculate equation (2), equation (3) is utilized.

For Iteration t = 1, . . . , t_{max}
  For each ant $k = 1, \ldots, m$
    Choose a city randomly
    For each non visited city i
      Choose a city j, from the list of J_i^k remaining cities using *(1)* given below.
    End For
    Deposit a trail $\Delta\tau_{ij}^k$ on the path $T^k(t)$ in accordance with *(3)* given below.
  End For
  Evaporate trails based on *(2)* given below.
End For

Figure 3.7 - An ACO algorithm used in solving the TSP

$$p_{ij}^k(t) = \begin{cases} \dfrac{(\tau_{ij}(t))^{\alpha}.(n_{ij})^{\beta}}{\sum_{l \in J_i^k}(\tau_{il}(t))^{\varepsilon}.(n_{ij})^{\beta}} & if \quad j \in J_i^k \\ 0 & if \quad j \notin J_i^k \end{cases} \tag{1}$$

where;

α and β are parameters that controls τ_{ij} and n_{ij}.

n_{ij} is the visibility from city i to city j.

τ_{ij} is the intensity from city i to city j.

ρ is evaporation coefficient.

where;

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{2}$$

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{Q}{L^k(t)} & if \quad (i,j) \in T^k(t) \\ 0 & if \quad (i,j) \notin T^k(t) \end{cases} \tag{3}$$

As the best solution is obtained, the algorithm stops. All the equations and algorithms are borrowed from Dréo et.al. (2006).

3.5 Greedy Randomized Adaptive Search Procedure (GRASP)

GRASP is another meta-heuristic method used for solving combinatorial optimization problems. Figure 3.8 demonstrates how GRASP works for a minimization problem.

This algorithm is composed of two main phases: a construction phase and a local search phase. In the construction phase, there is a greedy function which maintains the rankings of partial solutions. This step is very important because it affects the time efficiency of the algorithm. After ranking the partial solutions, some of the best ones are stored in a restricted candidate list (RCL). In the local search phase, as shown in Figure 3.8, a comparison is done

to differentiate the quality of solutions. The algorithm terminates after a fixed number of iterations.

```
procedure GRASP
    while (termination condition not met) do
        s ←ConstructGreedyRandomizedSolution
        ŝ ← LocalSearch(s)
        If f(ŝ) < f(s_best) then
            s_best ← ŝ
        end-if
    end-while
    return s_best
end-procedure
```

Figure 3.8 - High level pseudo-code for GRASP (Dorigo & Stützle, 2004)

Fogel & Michalewicz (2000) provide a GRASP application to solve a TSP with 70 cities. They randomly select a city to begin the tour and then add the other 69 cities one at a time to the tour. After constructing an initial solution, they run the algorithm and evaluate 2415 different solutions. In such big TSP problems, GRASP seems to find good solutions in reasonable amounts of time.

3.6 Particle Swarm Optimization (PSO)

PSO is inspired from the collective behaviors of animals. In this section, we will present a sample PSO algorithm to demonstrate how it works and talk about the kinds of problems it is applied to.

There are two key definitions in using PSO algorithms that have been defined in Section 2 earlier: position and velocity. The position and velocity of particle i at time t are represented by $x_i(t)$ and $v_i(t)$ respectively. The position and velocity of a particle changes based on the following equations:

$$\boldsymbol{x}_i(t) = \boldsymbol{x}_i(t-1) + \boldsymbol{v}_i(t-1) \tag{4}$$

equivalently, $\boldsymbol{x}_i(t)$ can be represented as a function of the previous position, previous velocity, $\boldsymbol{p}_i$, and $\boldsymbol{p}_g$ where, $\boldsymbol{p}_i$ is the local best position of particle i, and $\boldsymbol{p}_g$ is the neighborhood best position

$$\boldsymbol{x}_i(t) = f(\boldsymbol{x}_i(t-1), \boldsymbol{v}_i(t-1), \boldsymbol{p}_i, \boldsymbol{p}_g) \tag{5}$$

Equation (6) shows the velocity of particle i.

$$\boldsymbol{v}_i(t) = \boldsymbol{v}_i(t-1) + \Phi_1 (\boldsymbol{p}_i - \boldsymbol{x}_i(t-1)) + \Phi_2 (\boldsymbol{p}_g - \boldsymbol{x}_i(t-1)) \tag{6}$$

where;

Φ_1 and Φ_2 are randomly chosen parameters. Φ_1 represents the individual experience and Φ_2 represents the social communication. In figure 3.9 the PSO algorithm is given for n particles:

```
    For i = 1 to n :
        If F(x_i) > F(p_i) then :
            For d = 1, . . . , D :
                p_id = k_id // p_id is thus the best found individual
            end d
        end if
        g = i
        For j =index of the neighbors:
            If F(p_j) > F(p_g) then:
                g = j // g is the best individual in the
                neighborhood
            end if
        end j
        For d = 1, . . . , D :
                v_id(t) = v_id(t − 1) + Φ1 (p_id − x_id (t − 1)) + Φ2 (p_gd −
                - x_id (t − 1))
                v_id ∈ (−V_max_ + V_max)
                v_id(t) = x_id(t − 1) + v_id(t)
        end d
    end i
end
```

Figure 3.9 - The PSO algorithm for n particles (Dréo et al., 2006)

As seen in Figure 3.9, this algorithm can be used in multiple dimensions.
This PSO algorithm can be applied to many problems in the real life such as the TSP, the vehicle routing problem, the flowshop scheduling problem, etc. However, it is more commonly used in training of artificial neural networks.

4. Tabu search applications

TS applications comprise diverse fields like scheduling, computer channel balancing, cluster analysis, space planning, assignment, etc. It also has applications in many different technical problems like the travelling salesman, graph coloring, character recognition, etc. We have reviewed the recent literature (2000 and after) using keywords such as "tabu search", "local search", and "meta-heuristic." We collected around 150 articles most of which are focused on TS. In this chapter we will go over these articles to point out various applications of TS. Rather than going over all 150 articles, we only focused on the ones that represent diverse applications of TS.
Based on our literature review, TS is used widely on machine scheduling and job-shop scheduling problems. In his study Glover (1990), stated that Widmer & Hertz's (1990) application of TS to flow shop sequencing problems succeeded in obtaining solutions superior to the best previously found by applying a range of methods in about 90% of the cases. TS has shown superior results in other recent applications as well. Blazewicz et al. (2008) presented three meta-heuristics SA, TS, and variable neighborhood search (VNS) for the two-machine flow-shop problem with weighted late work criterion and common due date. Initial solutions were generated by Johnson's algorithm (1954) or list scheduling

algorithm which is a constructive method, that builds a solution by executing jobs selected according to a given priority dispatching rule. In order to have the best settings for the corresponding meta-heuristics, some parameters were tuned. As a result of the experiments TS control parameters were selected as: the neighborhood was generated by interchanging jobs based on the weighted processing times of the jobs, from 33% of generated jobs, termination condition was double the number of jobs, tabu list length was equal to 300% of the number of jobs. For comparing the efficiency of meta-heuristics, 20 problem instances were evaluated. Tuning improved TS performance by 7%. In this study, SA generated better schedules in shorter time. Chen et al. (2007) studied an extension of the hybrid flow-shop scheduling problem. In their study, a mixed-integer programming (MIP) model was provided, then a TS based algorithm was used. Also, Fink & Voss (2003) examined the application of different kinds of heuristic methods to the continuous flow-shop scheduling problem. Results show that effectiveness of heuristics depends on the problem size, desired solution quality, and available running time meaning that there is no single ''best'' method that dominates all other heuristics. In general, reactive TS (where tabu list is shortened or increased according to a resampling condition) obtained high-quality results without any parameter tuning. Brucker et al. (2003) presented a TS approach for the flowshop problem with intermediate buffers where different job sequences on the machines were allowed. Pan et al. (2008) proposed a PSO algorithm for a no-wait flowshop scheduling problem. When implementing the PSO algorithm, they also made comparisons of the performance of PSO with several other meta-heuristic procedures. One comparison was in between PSO and TS on this problem. It was mentioned that the TS algorithm and its hybrids generated better results on this specific problem. Eksioglu et.al. (2008) also used a TS algorithm utilizing changing neighborhoods for a flowshop scheduling problem. In their study, the results of this algorithm (3XTS) were compared to neuro-tabu search (EXTS) and ant colony optimization (ACO) algorithms. The property of the used TS algorithm in their research was that TS used three different neighborhood structures to obtain better results which meant it diversified the search. They observed that this TS algorithm obtained as good solutions as the neuro-TS and ACO algorithms. The computational time of 3XTS was almost the half of EXTS because it was capable to explore block properties. They concluded that the solution times of the ACO and the 3XTS algorithms were almost the same but 3XTS gived solutions which were closer to the optimal solution. Chen et al. (2008) developed a hybrid TS (HTS) for re-entrant permutation flow-shop scheduling problems. The proposed algorithm (HTS) improved the efficiency of TS obtaining favorable solutions within a reasonable time. It was mentioned that HTS found optimal solutions for all small problems. For large problems, HTS was superior to pure TS. Moreover, as the size of the problem increased, HTS performed better than pure TS.

Zhang et al. (2007) studied job-shop scheduling problems (JSP) proposing a TS algorithm with a new neighborhood structure that could avoid cycling and investigate much larger part of the solution space. Kis (2003) had also solved alternative JSP problems by TS and by a GA. Based on the results, TS was superior to GA both in terms of solution quality and computation time. Pezzella & Merelli (2000) presented a TS method guided by shifting bottleneck for the JSP. Liaw (2000) had developed a hybrid GA based on TS for open shop scheduling problem. It was stated that the algorithm performed extremely well Liaw (2003) examined the problem of scheduling two-machine preemptive open shops. A TS approach was proposed that provided excellent results. In most of the cases TS found optimum

values. In other cases it found values with average deviation from optimum value by nearly 2%. However, McMullen & Frazier (2000) dealt with the problem of mixed-model sequencing with multiple objectives on a just-in-time line. They used SA for solving the problem. It was stated that SA outperformed TS in most of the cases. Grabowski & Pempera (2000) dealt with sequencing of jobs in a production system presenting a TS a solution method. Vinicius et al. (2000) used TS for scheduling on identical parallel machines to minimize mean tardiness. Kim et al. (2003) also used a due date density-based categorizing heuristic that incorporated TS for parallel machines scheduling.

Waligóra (2008) studied discrete–continuous (discrete and continuous resources) project scheduling with discounted cash flows. Applications of TS, as well as simple search methods have been described. Based on the experiments, TS seemed to be an efficient algorithm for solving the considered problem, clearly outperforming simple search algorithms and producing results close to optimum. Mika et al. (2008) also studied a multi-mode resource-constrained project scheduling problem with schedule dependent setup times. TS was compared to a multi-start iterative improvement method and a random sampling method. Experiments showed that TS outperformed the other algorithms. Valls et al. (2003) dealt with the resource-constrained project scheduling problem through a non-standard implementation of TS principles.

Rubrico et al. (2008) studied scheduling of multiple picking agents for warehouse management. They developed a tabu scheduler exhibiting less computation time. Burke et al. (2004) proposed a TS hyperheuristic for timetabling and rostering. It outperformed GA in terms of feasibility while GA was better in terms of cost.

Burke & Smith (2000) developed a hybrid algorithm that was a memetic algorithm using TS for maintenance scheduling problem. It was concluded that this algorithm performed better at the expense of a little increase in solution time. Based on the results presented in the above references, we can conclude that a pure TS or hybrid TS performs superior to other advanced meta-heuristics like SA, GA or PSO and simple heuristics for flowshop scheduling, JSP and general scheduling problems.

Like scheduling, vehicle routing is another field in which TS is widely used. Teng et al. (2003), performed a comparative study of meta-heuristics for the vehicle routing problem (VRP) with stochastic demands. In this study, they present three meta-heuristics: SA, threshold accepting (TA), and TS meta-heuristic for the single VRP with stochastic demands. It is shown that quality of initial solutions has a positive effect on the TS algorithm while this is not valid for SA and TA. In this study, solution quality of TS outperforms the other two. Moreover, the superiority of TS increases as the problem size increases. Also TS requires less computation time than SA. Same neighborhood is used for all of the meta-heuristics. Hence, it is concluded that TS performs better than SA and TA in terms of solution quality and computation time for the single VRP with stochastic demands. Fallahi et al. (2008) introduced a multi-compartment vehicle routing problem (MC-VRP), a problem not yet studied in literature in spite of its important practical applications. Three algorithms were proposed to solve it: a constructed heuristic, a memetic algorithm (MA) combined with a path relinking method used as post optimization, and TS. These methods have been tested using two sets of problems. It is stated that in general, TS provides slightly better solutions than MA but requires more computational time. Li et al. (2007) compared the results of 11 algorithms that solve the open vehicle routing problem (OVRP) and found that procedures based on adaptive large neighborhood search, record-to-record travel, and TS performed

well. Shetty et al. (2008) considered the strategic routing of a fleet of unmanned combat aerial vehicles (UCAVs). The TS heuristic for TSP calculates quick tours when the assignments are finalized. For larger scale problems the TS heuristic provides good feasible solutions quickly. Iterative nature of the heuristic allows it to be stopped after a feasible solution is generated. It is stated that such problems might benefit from the decomposition scheme proposed in this work with TS coordinating the subproblems. Caricato et al. (2003) have examined a VRP under Track Contention whose applications arise in automated material handling systems and flexible manufacturing systems. Three heuristics were developed: two simple procedures and a tailored TS. In view of its adaptation to a real-time setting, the TS algorithm was parallelized. Based on the results, TS provides better solutions although this usually comes at the expense of a larger computing time. Backer et al. (2000) dealt with VRPs presenting a method for combining constraint programming with local search including TS. Gendreau et al. (2007) proposed a TS heuristic for the vehicle routing problem with two-dimensional loading constraints.

TS is used in facility location problems as well. Arostegui et al. (2006) compared relative performance of TS, SA, and GA on various facility location problems (FLP). In their study it is mentioned that another comparison between these meta-heuristics was made in 1993 by Sinclair for the quadratic assignment problem (QAP). In the FLP problem GA performed worst and TS provided better solutions than SA in 28 instances out of 37 having nearly the same computation time. In 1992 Kincaid compared TS and SA for noxious facilities location and concluded that TS outperformed SA. In 2003, Wang et al. showed that for the budget constraint location problem TS results were more satisfactory in terms of quality when compared to a Lagrangian Relaxation (LR) approximation. Also Chamberland in 2004 showed that for a network subsystem expansion problem a TS based heuristic provided good solutions. Capacitated (CFLP), multi-period (MPFLP), and multi-commodity (MCFLP) facility location problems were used to compare three meta-heuristics. Their performance was evaluated on three dimensions: computation time limitation, solution limited dimension, unrestricted dimension. For time-limited results, performance of TS was superior to others in CFLP and MPFLP. Statistically TS showed best performance for rapidly reaching low-cost solutions followed by SA and GA. For MCFLP, GA provided best result. Vector representation for MCFLP resulted in fewer solutions to be evaluated. Overall, given the same amount of time TS in general gave the best results. For solutions-limited results, SA performed similar to or better than TS. For unrestricted results, TS performed best for CFLP and MCFLP while SA performed best for MPFLP. Michel & Hentenryck (2004) presented a simple tabu-search algorithm which performed well for un-capacitated warehouse location problem (UWLP). It outperformed the GA in efficiency and robustness. Amaldi et al. (2006) investigated mathematical programming models for base station location and configuration. They proposed a TS algorithm starting with an initial solution using a randomized greedy procedure. Mladenovic et. al (2003) proposed a TS algorithm and variable neighborhood search (VNS) for *p*-Center problem, one of the basic models in discrete location theory. Cortinhal & Captivo (2003) dealt with the single source capacitated location problem. They proposed a Lagrangean heuristic combined with TS.

The Capacitated Arc Routing Problem (CARP) is another application for TS. Brandão and Eglese (2008) presented a deterministic TS algorithm for the CARP. Results in this study show that TS is capable of providing high quality results in a reasonable computing time. Amberg et al. (2000) dealt with multiple center CARP with a TS algorithm using capacitated

trees. They concluded that the proposed TS is capable of yielding good solutions. Brandao and Eglese (2006) solved a capacitated arc routing problem using a deterministic TS algorithm and they compared the results with a memetic algorithm and CARPET (their heuristic for solving CARP). They concluded that the results of the TS algorithm were reproducible because it was a deterministic algorithm. On the other hand, they denoted that the results of the memetic algorithm and CARPET could not be reproduced because these algorithms included random elements. Since the memetic algorithm and CARPET used random elements, everytime they run the algorithms they obtained different results so they preferred the results of the TS algorithm.

In the network design field, Ignacio et al. (2008) used TS in a computer network design problem called the concentrator location problem. A problem formulation and a LR procedure were presented. Approximate solutions are obtaind by TS. Computational results are shown for 320 problem instances. For problems with more than 100 users TS provides better results than CPLEX. Two other studies in network design were by Chamberland (2003) and Fortz et al. (2003).

In production and distribution field, Russell et al. (2008) studied a problem for integrating the production and distribution of newspapers from plant to bulk delivery locations. The proposed TS methodology made it easier and more efficient for the newspaper to handle increasing volume of pre-print advertisement. TS methodology was able to reduce the number of vehicles required by 18.18%. Lukac et. al. (2008) studied production planning problems with sequence dependent setups and solved them by a TS based heuristic. Valdes et al. (2007) studied a two-dimensional non-guillotine cutting problem proposing a TS algorithm. Based on the computational results, TS worked well for the constrained and double-constrained test problems. Onwubolu et al. (2000) proposed a TS approach to cellular manufacturing systems. In this study a cell formation problem was modeled that had three objectives: minimization of intercellular movements, minimization of cell load variation, and a combination of two. It was stated that TS worked as well as other published algorithms for the same problem. TS has an extra advantage that is allowing the designer to select the maximum number of cells as well as machines in a cell. Hung et al. (2003) used TS with ranking candidate list to solve production planning problems with setups. Pai et al. (2003) dealt with optimization of laminate stacking sequence for failure load maximization using TS. Results were comparable to GA.

There are many other application fields and problems in which TS is used. For example: Cell planning with capacity expansion in mobile communications (Lee & Kang, 2000), application-level synthesis methodology for multidimensional embedded processing systems (Alippi et al. 2003). Cogotti et al. (2000) performed a comparison of optimization techniques for Loney's Solenoids Design and proposed an alternative TS algorithm. Emmert et al. (2003) have shown an effective way of bi-partitioning electrical circuits using TS. It was stated that TS offered quick convergence to good partitioning solutions for circuits in the range of their application. Their algorithms show dramatic improvement in execution time with good solution quality as compared to a random move SA approach. They also mention that their placement method is suitable for quickly initializing the inputs to other non-deterministic placement algorithms. Rajan et al. (2003) proposed a neural-based TS method for solving unit commitment problem. Blazewicz et al. (2000) proposed a TS-based algorithm for DNA sequencing in the presence of false negatives and false positives. Corberan et al. (2000) studied a mixed rural postman problem in which TS was used. Ahr &

Reinelt (2005) presented a TS algorithm for the min-max Chinese postman problem. Tan et al. (2008) proposed an optimization procedure combining zonation methods with TS to identify the spatial distribution of a hydraulic conductivity field. Blöchliger & Zufferey (2008) studied a graph-coloring problem proposing a heuristic using partial solutions and a reactive tabu scheme. It was shown that this reactive tabu scheme obtains good results on a large sample of benchmark graphs which are generally difficult to color. Konak et al. (2003) studied a reliability design problem called the Redundancy Allocation Problem. A TS, named TSRAP,was described and compared to other approaches to the problem.
As amply demonstrated, TS is applied to a large variety of problems that arise in different fields. Next, we discuss performance of TS compared to other meta-heuristics.

5. Performance of tabu search

In 1988, the Committee on the Next Decade of Operations Research (CONDOR) evaluated TS together with SA and GA to be "extremely promising" for the future treatment of practical applications (Glover, Laguna, 1997). Based on section 4, we can state that the committee predicted the future quite well. TS, incorporating many artificial intelligence properties, outperformed other meta-heuristics in many application fields. However, theoretical aspects of TS that make it successful are still a matter of discussion. Some nice properties of TS are: TS generally proceeds more aggressively to local optimum unlike SA which relies on the premise that a slow descent will lead to a local optimum that is closer to a global one. This rationale of TS derives from two considerations: (1) optimization problems can be solved making the best available move at each iteration; (2) rather than spending more time in regions whose solutions are less attractive TS devotes larger effort to exploring regions where solutions are good (Glover, 1989). SA may not be stopped at any desired moment in time since the control parameter (temperature) has to converge to a value close to zero to obtain a meaningful implementation, the cooling schedule needs to be tuned to the time available for deriving a solution (Michiels et al., 2007). TS can be stopped at any time. Iterative nature of the TS may allow it to be stopped after a feasible solution is found. GA is a population based approach. It requires evaluation of populations over generations. For complex problems that kind of approach results in a great computational effort. TS, considering neighboring moves and not needing objective function gradient information as GA, is more efficient as demonstrated by Konak et al. (2003). SA makes stochastic moves. However TS uses deterministic moves which reduces variability due to initial solutions and other parameters.
However, it is obvious that for a successful implementation of TS it is necessary to tune the algorithm for the specific problem on hand. Jaeggi et al. (2008) states that "A major shortcoming in the assessment of optimization algorithms for use on real-world problems is the lack of a suitable set of benchmark problems, which accurately capture the main features of the problems of interest. Until such a set is developed, true performance comparison between algorithms is difficult and one must rely on inference from a less suitable set of benchmark problems." Based on the experiments discussed in our review of the literature, tabu lists designed to prevent repetition rather than reversal of moves seem to not work well. An empirical discovery for the application of TS methods is that the number of iterations has a highly stable range of values that prevents both cycling and leads to good solutions (Glover, 1989). Development of TS is an iterative process, thus hoping to find the best solution at the very beginning would be naive. Necessary modifications depending on the problem instance must be made.

As mentioned in (Dréo et al. 2006) for many applications TS based heuristics showed more effective results. TS owes its efficiency to rather fine tuning of a large collection of parameters which may seem counter-intuitive. TS is a trajectory method constituting a neighborhood of solutions at each iterations in contrast to SA generating a single solution. In some cases, that attribute of TS makes it slower than SA as shown by Blazewicz et al. (2008). Deterministic TS may be feasible to be able to reproduce the results as shown by Brandão & Eglese (2008). GA appears to perform well in an environment when information is limited as in MCFLP. It seems like the longer the solution time the better the probability that TS will show superior performance. Local-search component and constraint handling flexibility of TS makes it attractive for problems having many constraints (Jaeggi et al., 2008). Vallada et al. (2008) stated that from the meta-heuristics tested, the two SA algorithms proposed outperform all other methods evaluated. They have also shown that the TS methods are good meta-heuristics in the *m*-machine flowshop problem with the objective of minimizing total tardiness. The method used for intensification by TS or GA, plays an important role in determining the accuracy of the results. Liaw (2000) stated that GA is good at performing global search and TS is effective for fine tuning for the open shop scheduling problem. A hybrid approach combining GA and TS in that study found optimal solution for nearly all test problems. Incorporation of appropriate heuristics with pure TS may be more effective as shown by Chen, et al. (2008). If the evaluation of the entire neighborhood space requires too much computation, then a neighborhood sorting method as well as using a ranking candidate list strategy may improve the performance of TS (Hung et al., 2003).

6. Conclusion

In today's global and competitive environment, the use of scarce resources in the best possible way is more important than ever, and time is one of the critical resources for almost all problems. In this book chapter, we tried to show how efficient and effective results can be obtained using meta-heuristic methods, specifically TS.

According to editorial of European Journal of Operational Research (EJOR) (Editorial, 2007), meta-heuristic research efforts seem to be aimed at two main areas of application: production/scheduling problems and logistics problems. Other domains of applications in a related issue of EJOR are finance, product design, bio-computing and data mining. A common feature seen is that authors use hybrid meta-heuristics to acquire efficient tailor-made solution approaches.

Hybrid approaches seem to have better performance in some problems motivating researchers to focus on hybrid meta-heuristic usage. For example, Ting et al. (2003) presented a work focusing on a novel mating strategy, called tabu genetic algorithm (TGA). TGA integrates tabu search (TS) into GA's selection. Structures of GA and TS are not modified in these approaches. It is shown that in contrast to running GA and TS alternately, TGA implants characteristics of TS like aspiration into GA's mating strategy. It is concluded that TGA outperforms GA, TS, and conventional hybrids of GA and TS, in terms of solution quality and convergence speed.

Glover (2007) mentions that from the beginning of TS journey to present enabled us to solve many kinds of optimization problems effectively although there is a long way to go. Glover suggests focusing on human memory usage to understand how it affects problem solving, so that these features can be incorporated to TS memory. For this case psychology and heuristics fields should engage in new projects.

7. References

Aarts, E. & Lenstra, J., K. (2003). *Local Search in Combinatorial Optimization*, Princeton University Press, 0-691-11522-2, New Jersey

Ahr, D. & Reinelt G. (2005). A tabu search algorithm for the min–max k-Chinese postman problem, *Computers and Operations Research*, 33 (7 December 2005), 3403-3422, 0305-0548.

Alippi, C., Galbusera A., & Stellini M. (2003). An Application-Level Synthesis Methodology for Multidimensional Embedded Processing Systems, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 22, No. 11, 0278-0070

Alves, M. J. & Climaco, J. (2000). an Interactive Method for 0-1 Multiobjective Problems Using Simulated Annealing and Tabu Search, *Journal of Heuristics*, 6, 2000, 385–403

Amaldi, E.; Belotti, P.; Capone, A. & Malucelli, F. (2006). Optimizing Base Station Location and Configuration, *Ann Oper Res*, 146, 27 June 2006, 135–151

Amberg, A.; Domschke, W. & Voss S. (1999). Multiple Center Capacitated Arc Routing Problems: A Tabu Search Algorithm Using Capacitated Trees, *European Journal of Operational Research,* 124, 2000, 360-376, 0377-2217

Armentano, V.A. & Yamashita D.S. (2000). Tabu search for scheduling on identical parallel machines to minimize mean tardiness, *Journal of Intelligent Manufacturing*, 453-460 Arostegui, M., A.; Kadipasaoglu, S., N. & Khumawalab, B., M. (2006), an Empirical Comparison of Tabu Search, Simulated Annealing, and Genetic Algorithms for Facilities Location Problems. *Int. J. Production Economics*, 103, 2006, 742–754, 0925-5273

Backer, B.E. & Furnon, V. & Shaw, P. (2000). Solving Vehicle Routing Problems Using Constraint Programming and Meta-heuristics, *Journal of Heuristics,* 501-523

Blazewicz, J.; Formanowicz, P.; Kasprzak, M.; Markiewicz, W., T. & Weßglarz, J. Tabu Search for DNA Sequencing with False Negatives and False Positives, *European Journal of Operational Research,* 125, 2000, 257-265, 0377-2217

Blazewicza, J.; Peschb,E.; Sternaa,M. & Wernerc F. (2006). Meta-heuristic Approaches for the Two-Machine Flow-Shop Problem with Weighted Late Work Criterion and Common Due Date. *Computers & Operations Research,* 35, 2008, 574 – 599, 0305-0548

Blöchligera, I. & Zuffereyb, N. (2006). a Graph Coloring Heuristic Using Partial Solutions and a Reactive Tabu Scheme, Computers & Operations Research, 35, 2008, 960 – 975, 0305-0548

Blum, C. & Roli, A. (2003). Meta-heuristics in Combinatorial Optimization: Overview and Conceptual Comparison, *ACM Computing Surveys,* Vol. 35, No. 3, September 2003, 268–308Brandão, H. & Eglese, R. (2006). A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers and Operations Research,* 35 (25 September 2006),1112-1126, 0305-0548

Brandãoa, J. & Egleseb R. (2006). a Deterministic Tabu Search Algorithm for the Capacitated Arc Routing Problem. *Computers & Operations Research,* 35, 2008, 1112 – 1126, 0305-0548

Brucker, P.; Heitmann, S. & Hurink, J. (2003). Flow-shop problems with intermediate buffers, *OR Spectrum,* 25, 2003, 549–574

Budenbender, K.; Grunert, T. & Sebastian, H. (2000). a Hybrid Tabu Search/Branch-and-Bound Algorithm for the Direct Flight Network Design Problem, *Transportation Science,* 34, 4, November 2000, 364–380, 1526-5447

Burke, E., K. & Smith A. J. (2000). Hybrid Evolutionary Techniques for the Maintenance Scheduling Problem, *IEEE Transactions On Power Systems*, 15, 1, February 2000, 122-128, 0885-8950

Burke, E.K., Kendall, G. & Soubeiga, E. (2004). A Tabu-Search Hyperheuristic for Timetabling and Rostering, *Journal of Heuristics,* August 2003,451-470

Caricato, P.; Ghiani, G.; Grieco, A. & Guerriero, E. (2002). Parallel Tabu Search for a Pickup and Delivery Problem under Track Contention, *Parallel Computing,* 29, 2003, 631–639, 0167-8191

Chamberland, S. (2003). on the Overlay Network Design Problem for the Soft-Label Switched Paths in IP Networks, *INFOR,* 41, 4, Nov. 2003, 301-332

Chelouah, R. & Siarry, P. (2000). Tabu Search Applied to Global Optimization. *European Journal of Operational Research,* 123, 2000, 256-270, 0377-2217

Chen, J.; Pan, J., C. & Wu, C. (2008). Hybrid Tabu Search for Re-Entrant Permutation Flow-Shop Scheduling Problem, *Expert Systems with Applications,* 34, 2008, 1924–1930, 0957-4174

Chen, Lu.; Bostel, N.; Dejax P.; Cai J. & Xi L. (2006). a Tabu Search Algorithm for the Integrated Scheduling Problem of Container Handling Systems in a Maritime Terminal, *European Journal of Operational Research*, 181, 2007, 40–58, 0377-2217

Cogotti, E.; Fanni, A. & Pilo, F. (2000). A Comparison of Optimization Techniques for Loney's Solenoids Design: An Alternative Tabu Search Algorithm, *IEEE Transactions On Magnetics,* 36, 4, July 2000, 1153-1157, 0018–9464

Corberan, A.; Marti, R. & Romero, A. (1998). Heuristics for the Mixed Rural Postman Problem, *Computers & Operations Research*, 27, 2000, 183-203, 0305-0548

Cortinhal, M.J. & Captivo, M.E. (2003). Upper and lower bounds for the single source capacitated location problem, *European Journal of Operations Research,* 333-351, 0377-2217

Dorigo, M. & Stützle, T. (2004). Ant Colony Optimization, *The MIT Press,* ISBN 0-262-04219-3, United States of America

Dréo, J.; P´etrowski, A.; Siarry, P. & Taillard, E. (2006). *Meta-heuristics for Hard Optimization,* Springer-Verlag Berlin Heidelberg, 10 3-540-23022, New York

Editorial, (2007). Applications of Meta-heuristics. *European Journal of Operational Research*, 179, 2007, 601–604, 0377-2217

Eksioglu, B., Eksioglu, S. D. and Jain, P. (2008). A tabu search algorithm for the flowshop scheduling problem with changing neighborhoods, *Computers and Industrial Engineering,* 54 (13 April 2007),1-11, 0360-8352

Emmert, J., M.; Lodha, S. & Bhatia, D., K. (2003). On Using Tabu Search for Design Automation of VLSI Systems, *Journal of Heuristics*, 9, 2003, 75–90

Fallahi, A.; Prins, C. & Calvo, R., W. (2006). a Memetic Algorithm and a Tabu Search for the Multi-Compartment Vehicle Routing Problem, *Computers & Operations Research*, 35, 2008, 1725 – 1741, 0305-0548

Fink, A. & Voß S. Solving the Continuous Flow-Shop Scheduling Problem by Meta-heuristics, *European Journal of Operational Research*, 151, 2003, 400–414, 0377-2217

Fortz, B.; Soriano, P. & Wynants, C. (2003). a Tabu Search Algorithm for Self-Healing Ring Network Design, *European Journal of Operational Research*, 151, 2003, 280–295, 0377-2217

Gendreau, M., Lori, M., Laporte, G. & Martello, S. (2007). A Tabu Search Heuristic for the Vehicle Routing Problem with Two-Dimensional Loading Constraints, *Interscience*, Vol. 51(1), October 2005, 4-18

Glover, F. & Laguna, M. (1997). Tabu Search, *Kluwer Academic Publishers*, 0-7923-8187-4, Massachusetts

Glover, F. (1988). Tabu Search – Part I, *ORSA Journal on Computing*, Vol.1, No.3, Summer 1989, 0899-1499/89/0103-0190

Glover, F. (1989). Tabu Search – Part II, *ORSA Journal on Computing*, Vol.2, No.1, Winter 1990, 0899-1499/90/0201-0004

Glover, F. (2006). Tabu search—Uncharted Domains, *Ann. Oper. Res.* 149, 2007, 89–98

Grabowski, J. & Pempera, J. (2000). Sequencing of jobs in some production system, *European Journal of Operations Research,* 1 March 1999,535-550, 0377-2217

Hasan, M.; AlKhamis, T. & Ali J. (2000). a Comparison between Simulated Annealing, Genetic Algorithm and Tabu Search Methods for the Unconstrained Quadratic Pseudo-Boolean function, *Computers & Industrial Engineering*, 38, 2000, 323-340, 0360-8352

Hertz, A. & Werra, D. (1990). The Tabu Search Meta-heuristic: How We Used It, *Annals of Mathematics and Artificial Intelligence,* 1, 1-4, September (1990), 111-121, 1573-7470

Hindsberger, M. & Vidal, R. (2000). Tabu Search – A Guided Tour, *Control and Cynernetics*, 29, 3, 2000

Holder, A. editor. Mathematical Programming Glossary. *INFORMS Computing Society*,

http://glossary.computing.society.informs.org/, 2006-2007. Originally authored by Harvey J. Greenberg, 1999-2006.

Hung, Y.F., Chen, C.P., Shih, C.C. & Hung, M.H. (2003). Using tabu search with ranking candidate list to solve production planning problems with setups, *Computers & Industrial Engineering,* 11 September 2003, 615-634), 0360 8352

Ignacio, A., A., V.; Filho, V., J., M., F. & Galvão R., D. (2006). Lower and Upper Bounds for a Two-Level Hierarchical Location Problem in Computer Networks, *Computers & Operations Research*, 35, 2008, 1982 – 1998, 0305-0548

Imahori, S., Yagiura, M., & Ibaraki T. (2003). Local search algorithms for the rectangle packing problem with general spatial costs, *Springer,* November 25 2002, 543-569

Jaeggi, D. M., Parks, G.T., Kipouros, T. & Clarkson P.J. (2006), The development of a multi-objective Tabu Search algorithm for continuous optimisation problems. *European Journal of Operational Research,* 185 (25 October 2006),1192-1212, 0377-2217

Jaeggi, D., M.; Parks, G., T.; Kipouros, T. & Clarkson P.J. (2006). the Development of a Multi-Objective Tabu Search Algorithm for Continuous Optimisation Problems, *European Journal of Operational Research*, 185, 2008, 1192–1212, 0377-2217

Jiang, T.; Luo, A.; Li, X. & Kruggel, F. (2003). a Comparative Study of Global Optimization Approaches to Meg Source Localization, *Intern. J. Computer Math.,* 80(3), 2003, 305–324, 1029-0265

Johnson, S. (1954). Optimal Two and Three Stage Production Schedules with Set-up Times Included, *Naval Research Logistics Quarterly,* 1, 1, 1954, 61-68

Kidwai, F.A., Marwah B.R., Deb, K. & Karim M. R. (2005). A Genetic Algorithm Based Bus Scheduling Model for Transit Network, *Proceedings of the Eastern Asia Society for Transportation Studies,* Vol. 5,477 – 489

Kim, S.S., Shin H.J., Eom, D.H. & Kim, C.O. (2003). A due date density-based categorising heuristic for parallel machines scheduling, *International Journal of Advanced Manufacturing Technology,* 26 July 2003,753-760

Kis, T. (2003)Job-shop scheduling with processing alternatives European Journal of Operational Research, 151, 2003, 307–332, 0377-2217

Konak, S.;Smith, A. & Coit, D. (2003). Efficiently Solving the Redundancy Allocation Problem Using Tabu Search, *IIE Transactions*, 35, 515-526, 2003

Laurent, M. & Hentenryck, P., V. (2004). a Simple Tabu Search for Warehouse Location European Journal of Operational Research, 157, 2004, 576–591, 0377-2217

Lee, C., Y. & Kang, H., G. (2000). Cell Planning with Capacity Expansion in Mobile Communications: A Tabu Search Approach, *IEEE Transactions On Vehicular Technology,* 49, 5, September 2000, 1678-1691, 0018–9545

Lia, F.; Goldenb, B. & Wasilc, E. (2006). the Open Vehicle Routing Problem: Algorithms, Large-Scale Test Problems, and Computational Results, *Computers & Operations Research,* 34, 2007, 2918 – 2930, 0305-0548

Liaw, C. (1999). a Hybrid Genetic Algorithm for the Open Shop Scheduling Problem, *European Journal of Operational Research,* 124, 2000, 28-42, 0377-2217

Liaw, C.F. (2003). An efficient tabu search approach for the two-machine preemptive open shop scheduling problem, *Computers and Operations Research,* 1 September 2002, 2081-2095, 0305-0548

Lukac, Z., Soric, K. & Rosenzweig, V., V. (2006). Production Planning Problem with Sequence Dependent Setups as a Bilevel Programming Problem, *European Journal of Operational Research,* 187, 2008, 1504–1512, 0377-2217

McMullen, P.R. & Frazier, G.V. (2000). A simulated annealing approach to mixed-model sequencing with multiple objectives on a just-in-time line, 01 August 2000, *IIE Transactions,* 679-686

Merz, P. & Freisleben, B. (2000). Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem, *IEEE Transactions on Evolutionary Computation,* Vol. 4, No. 4 (November, 2000) 16 pages (337-352), 1089–778X

Michalewicz, Z. & Fogel, D.B. (2000). *How to Solve it : Modern Heuristics*, Springer, ISBN 3-540-22494-7, Germany.

Michiels W.; Aarts, E. & Korst, J. (2007). *Theoretical Aspects of Local Search*, Springer-Verlag Berlin Heidelberg, 3-540-35853-6, New York

Mika M., Waligóra, G.& Węglarz J. (2006). Tabu search for multi-mode resource-constrained project scheduling with schedule-dependent setup times. *European Journal of Operational Research,* 187 (15 November 2006), 1238-1250, 0377-2217

Mika, M.; Waligora G. & Weglarz, J. (2005). Tabu Search for Multi-Mode Resource-Constrained Project Scheduling with Schedule-Dependent Setup Times, *European Journal of Operational Research,* 187, 2008, 1238–1250, 0377-2217

Mladenović, N., Labbé, M. & Hansen, P. (2003). Solving the p-Center Problem with Tabu Search and Variable Neighborhood Search, *Networks,* Vol. 42, No. 1, 48-64

Mladenović, N., Petrović, J., Kovačević-Vujčić, V. & Ćangalović, M. (2003). Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search, *European Journal of Operations Research,*389-399, 0377-2217

Onwubolu, G. & Songore, V. (2000). a Tabu Search Approach to Cellular Manufacturing Systems, *Production Planning & Control*, 11, 2, 2000, 153–164, 0953-7287

Pai, N., Kaw, A. & Weng, M. (2003). Optimization of laminate stacking sequence for failure load maximization using Tabu search, *Composites,* 27 August 2002,405-413, 1359-8368

Palubeckis, K. (2004), Multistart Tabu Search Strategies for the Unconstrained Binary Quadratic Optimization Problem. *Annals of Operations Research,*259-282

Pan, Q. K., Tasgetiren, M.F. & Liang, Y.C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem, *Computers & Operations Research,*2807-2839, 0305-0548

Perez, J.; Moreno-Vega, J., M. & Martin, I., R. (2003). Variable Neighborhood Tabu Search and Its Application to the Median Cycle Problem, European Journal of Operational Research, 151, 2003, 365–378, 0377-2217

Pezzella, F. & Merelli E. (2000). A tabu search method guided by shifting bottleneck for the job shop scheduling problem, *European Journal of Operations Research,* 1 September 1998,297-310, 0377-2217

Rajan, C.C.A., Mohan M.R. & Manivannan, K. (2003). Neural-based tabu search method for solving unit commitment problem, *IEE Proceedings Online,* Vol. 150, No. 4, July 2003,469-475

Rubrico, F.I.U., Ota, F., Higashi, T. & Tamura, H. (2008). Meta-heuristic scheduling of multiple picking agents for warehouse management, *Industrial Robot: An International Journal,*58-68, 0143-991X

Russella, R.; Chianga, W. & Zepedab, D. (2006). Integrating Multi-Product Production and Distribution in Newspaper Logistics, *Computers & Operations Research*, 35, 2008, 1576 – 1588, 0305-0548

Shetty, V., K.; Sudit, M. & Nagi R. (2006). Priority-Based Assignment and Routing of a Fleet of Unmanned Combat Aerial Vehicles, *Computers & Operations Research*, 35, 2008, 1813 – 1828, 0305-0548

Tan, C.C., Tung, C.P. & Tsai, F.T.C. (2008). Applying zonation methods and tabu search to improve the ground-water modeling, *Journal of the American Water Resources Association,* May 2007,107-120

Teh, Y., S.; Rangaiah, G., P. Tabu Search for Global Optimization of Continuous Functions with Application to Phase Equilibrium Calculations, *Computers and Chemical Engineering,* 27, 2003, 1665-1679, 0098-1354

Teng, S.; Ong, H., L. & Huang, H., C. (2003). a Comparative Study of Meta-heuristics for Vehicle Routing Problem with Stochastic Demands, *Asia-Pacific Journal of Operational Research*, 20, 2003, 103-119

Ting, C.; Li, S. & Lee, C. (2002), On the Harmonious Mating Strategy through Tabu Search, *Information Sciences*, 156, 2003, 189–214, 0020-0255

Valdes, R., A., Parreno, F. & Tamarit, J., M. (2006). a Tabu Search Algorithm for a Two Dimensional Non-Guillotine Cutting Problem, *European Journal of Operational Research*, 183, 2007, 1167–1182, 0377-2217

Vallada E.; Ruiz, R. & Minella, G. (2006). Minimising Total Tardiness in the M-Machine Flowshop Problem: A Review and Evaluation of Heuristics and Meta-heuristics, *Computers & Operations Research*, 35, 2008, 1350 – 1373, 0305-0548

Valls, V., Quintanilla, S. & Ballestín F. (2003). Resource-constrained project scheduling: A critical activity reordering heuristic, *European Journal of Operations Research,* 282-301, 0377-2217

Waligóra, G. (2006). Discrete–Continuous Project Scheduling with Discounted Cash Flows—A Tabu Search Approach, *Computers & Operations Research*, 35, 2008, 2141 – 2153, 0305-0548

Waligóra, G. (2008). Discrete–continuous project scheduling with discounted cash flows—A tabu search approach, *Computers & Industrial Engineering,* 27 October 2006, 2141-2153, 0305-0548

Watson, J.; Beck, J., C.; Howe, A. & Whitley, L. D. (2002). Problem Difficulty for Tabu Search in Job-Shop Scheduling, *Artificial Intelligence*, 143, 2003, 189–217, 0004-3702

Zhang, C., Y.; Li, P.; Guan, Z. & Rao, Y. (2006). a Tabu Search Algorithm with a New Neighborhood Structure for the Job Shop Scheduling Problem, *Computers & Operations Research*, 34, 2007, 3229 – 3242, 0305-0548