5-1-2014

A Survey of Tabu Search in Combinatorial Optimization

Lemasri Piniganti
University of Nevada, Las Vegas

A SURVEY OF TABU SEARCH IN COMBINATORIAL OPTIMIZATION

By

Lemasri Piniganti

Bachelor of Technology, Information Technology
Jawaharlal Nehru Technological University, India
2011

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science - Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
May 2014

UNLV
UNIVERSITY OF NEVADA LAS VEGAS

THE GRADUATE COLLEGE

We recommend the thesis prepared under our supervision by

Lemasri Piniganti

entitled

A Survey of Tabu Search in Combinatorial Optimization

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Department of Computer Science

Wolfgang Bein, Ph.D., Committee Chair

Ajoy K. Datta, Ph.D., Committee Member

Juyeon Jo, Ph.D., Committee Member

Venkatesan Muthukumar, Ph.D., Graduate College Representative

Kathryn Hausbeck Korgan, Ph.D., Interim Dean of the Graduate College

May 2014

ABSTRACT

A Survey of Tabu Search in Combinatorial Optimization

by

Lemasri Piniganti

Dr. Wolfgang Bein, Examination Committee Chair

Professor of Computer Science

University of Nevada, Las Vegas

Tabu search is a Meta heuristic loosely connected to evolutionary computing. It has been used to tackle hard problems, especially combinatorial optimization problems. Tabu search is designed to overcome difficult regions of a search space by imposing restrictions. Various methods for diversification and intensification are applied depending on the particular problem type and on what type of solutions (within the set of good solutions) are sought. Tabu search uses memory – short term, long term and intermediate – to achieve diversification and intensification. Furthermore, aspiration criteria may be used to tune the optimization process.

Thus the Tabu search Meta heuristic is very general. Different variants of the Tabu search Meta heuristic are presented in the context of combinatorial optimization. Problems discussed include the travelling salesman problem, various graph problems, and scheduling.

ACKNOWLEDGEMENTS

I would like to thank Dr. Wolfgang Bein, for being the chair committee and helping me in guiding throughout the period of my thesis. I am thankful to Dr. Bein for advising me this topic and helping me in providing the necessary papers for the successful completion of my work. It gives me great pleasure in expressing my gratitude to Dr. Bein.

I would also like to convey my special thanks to my graduate coordinator, Dr. Ajoy K. Datta for his support and guidance through my master's program. I am grateful to Dr. Datta for giving me chances to rectify my mistakes. I would like to thank my committee members Dr. Ajoy K. Datta, Dr. Ju-Yeon Jo and Dr. Venkatesan Muthukumar for giving their time and assessing my work. I would like to extend my thanks to the faculty of Computer Science and also for funding my master's program.

I would like to thank my sister for being the strength and support for me. I am also thankful to my friends who supported me with their presence in my life. I am grateful to my parents and sister for financially supporting me and giving me their immense love and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In the present day to day life, technology and information are increasing rapidly giving rise to the complexity of finding an optimal solution. To find a solution to a discrete problem with the presence of constraints or decision variables is called combinatorial optimization. This problem has been researched from the past 50 to 60 years. In order to solve this problem, meta-heuristic concept is introduced. In this thesis, a meta-heuristic called Tabu Search is introduced, and discusses the features of the tabu search algorithm. This is one of the most efficient heuristic in finding 'quality solutions' in relatively short running time. The principal characteristic of tabu search is based on using a mechanism which is inspired by the human memory [1] i.e., to use the information that is stored in the memory to guide and restrict the future search in a way to obtain quality solutions and to overcome the local optimality [5]. This research provides insight about the algorithm or procedure of the working of tabu search algorithm on combinatorial optimization problems like Travelling salesman problem, Job shop scheduling problem.

1.1 OUTLINE

Chapter 2 discusses the definitions of combinatorial optimization. It also provides the information about P and NP classes and also discusses about NP complete problems. It provides the information of the origin of tabu search and its definition. It also describes the heuristics and meta-heuristics. In chapter 3, the features of the tabu search are discussed. It provides the information about different aspects used in tabu search for achieving desired outcome based on the requirements of the problem. Different strategies

for intensification and diversification of the search are discussed in this chapter. Chapter 4 shows how the basic tabu search algorithm works, and it also gives the flow chart of the algorithm. Chapter 5 provides the applications of tabu search. In this research, graph problem, travelling salesman problem and job shop scheduling problem are used as examples to explain the tabu search algorithm. Chapter 6 presents the conclusion and its importance in the present field and the scope for improvements in the future.

CHAPTER 2

BACKGROUND AND LITERATURE

This chapter gives an insight about combinatorial optimization, meta-heuristics; Tabu search, its description and background of tabu search.

2.1 Optimization Problems

Optimization problem is often expressed as the problem to be solved and it is often supplemented by the information of constraints [4]. Optimizing can be defined as finding the maximum or minimum for an objective function defined on some domain. Finding optimal solutions depend on the objective function and constraints. If the objective function is too wild, the constraints too complicated or the problem size is too large, finding optimal solution is impossible [3]. This is the main concept for the theory of NP-Completeness.

Optimization problems can be distinguished into two types depending on the type of variables of the problem. They are problems with ‘discrete’ variables and problems with ‘continuous’ variables [2]. Travelling salesman problem is one of the problems that can be discussed among the discrete problems. An example for a continuous problem is that of the search for the values to be assigned to the parameters of a digital model of the process, so that this model reproduces the real behavior observed, as accurately as possible [3]. In practical, the problem can be comprised of both discrete variables and continuous variables.

Many researches are carried out for a long time to solve these two types of problems and found significant methods but these methods are effective only to a particular structural

property or specific to a given problem. ‘Meta-heuristics’ is a method that can be adapted to both kinds of optimizations.

2.1.1 Combinatorial Optimization

‘Combinatorics’ is a branch of mathematics studying the enumeration, combination, and permutation of sets of elements and the mathematical relations that characterize their properties. This definition is taken from Mathworld. Combinatorial optimization deals with the methods for optimizing the problems with ‘discrete variables’. The initial problems that are categorized as combinatorial optimization is the planning and management of operations and the efficient use of resources [3]. However at present there are so many combinatorial optimizations like, scheduling of production, sequencing of machines, transportation planning, design of unbreakable codes, etc. Combinatorial optimization is applied for a wide range of fields like in areas of sports, psychology, archeology, etc. [3]. Some of the problems in these areas are solved within a polynomial time, and some problems are difficult to get solutions in a considerable amount of time [6].

2.1.2 P and NP

A search problem is defined as there are different types of search problems in this world in which some of them can be solved efficiently, while others seem to be hard. So these problems can be differentiated as hard problems (NP Complete) and easy problems (P). The problem can be defined as both hard problem and easy problem by depending on its range of information.

The defining characteristic of a search problem is that, for any proposed solution, the correctness of the solution can be checked in a short interval of time. If the answer to the problem is defined as yes/no, based on the values of input parameters then that problem is known as a decision problem.

2.1.2.1 Class P

P stands for 'polynomial', so the collections of all problems that can be solved in polynomial time are listed into class P [7]. The mathematical way of defining this is, the given problem is in class P if there exist an exponent k and an algorithm for the problem that runs in time $O(n^k)$ where n is the length of the input [9]. In general class P contains practically solvable problems. Class P contains all decision problems that can be solved efficiently by using polynomial time algorithms.

Some of the problems which are classified into P class are,

2SAT, HORN SAT
MINIMUM SPANNING TREE
BIPARTITE MATCHING
SHORTEST PATH
UNARY KNAPSACK
LINEAR PROGRAMMING
EULER PATH

2.1.2.2 Class NP

NP stands for ‘nondeterministic polynomial’. The problem is classified into class NP, if the problem is solved in polynomial time by a nondeterministic Turing machine [9]. Class NP contains all the problems in which ‘verifying’ the solution of the problem is quick, but finding the solution for the problem is difficult. P is a subset of NP.

Some of the problems which are classified into NP class are,

3SAT
TRAVELLING SALESMAN PROBLEM
3D MATCHING
LONGEST PATH
KNAPSACK
INTEGER LINEAR PROGRAMMING
RUDRATA PATH
The major unsolved problem in Computer Science is P versus NP problem.

2.1.2.3 NP Hard

According to computational complexity theory, NP Hard (Non-deterministic Polynomial-time hard) is a class of problems that are at least as hard as the hardest problems in NP [7]. In a lot of cases, the problem can be solved by reducing it into a different problem. i.e., if there is a solution to problem A, then constructing the solution to problem B is easier [4].

If the problem is NP hard, then any problem in NP class can be reduced to that problem. So it means if that problem is solved, then any problem that are described in NP class can be solved easily.

Problems in NP hard do not have to be decision problems or in NP class.

The widely discussed and researched problem P = NP can be proved if there is a polynomial algorithm for any NP Hard problem.

Some examples of NP Hard Problem are Subset Sum problem, Halting problem, Travelling Salesman Problem, Graph isomorphism problem.

2.1.2.4 NP Complete

NP Completeness concept was presented by Stephen Cook in 1971 in a paper entitled The Complexity of Theorem-Proving Procedures. This paper was presented in a computer science conference for the proceedings of the 3rd annual ACM Symposium on Theory of Computing [9]. NP Complete is a specific case of NP class. The problem is said to be classified as NP Complete if it satisfies two conditions.

- The problem should be in the set of NP class.
- The problem should be NP hard.

NP Complete is a class of decision problems. There is no efficient way to find a solution for NP Complete problems but verifying a solution to this problem is comparatively easy. If the problem size is increased, then finding a solution with the present algorithms increases linearly with respect to the size of the problem, i.e., if the problem is even of moderate size the computational time increases to billion or trillion of years. So at present

most of the NP Complete problems are often dealt with approximation algorithms. NP Complete is a subset of NP class and all NP Complete problems are NP hard, but not all NP hard problems are NP Complete.

Some of the examples of NP Complete problems are 3-SAT, Sub graph isomorphism problem, Travelling Salesman Problem, Knapsack problem.

2.2 Heuristics

The term heuristic is derived from the Greek 'Heuriskein' meaning to find or discover. In the context of combinatorial optimization, the term heuristics is used as a contrast to methods that guaranteed to find a global optimum. A heuristic is a technique which seeks good solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is [2][6]. Heuristic is a procedure for finding optimal solutions but does not guarantee to find an optimal solution or quality of the solution even if one exists. Heuristics should be developed to deal with large problems and mostly a well-developed heuristic can at least give near optimal solutions.

Many researches are carried out on heuristic methods for solving the combinatorial problems, but most of these heuristics are problem-specific, so that a process that works for one problem cannot be used to solve a different one. So the hybrid methods, which endeavors to benefit from the specific advantages of different approaches by combining them are emerged, which can be applied far more generally.

2.2.1 Meta-heuristics

The word ‘Meta-heuristics’ is first used in the paper where the term tabu search is introduced (Glover, 1986). It is the general framework of heuristics in providing general structure and strategy guidelines to fit a particular problem. One main aspect of the meta-heuristic is to escape the local optima.

A meta-heuristic is a master strategy for guiding, modifying and controlling other heuristics to provide solutions beyond the usually generated solutions by some local heuristics. The meta-heuristics that use special procedures in order to not get trapped in local optima are one of the meta-heuristics. This meta-heuristic performs single search in the neighborhood. Tabu search is an example for this kind of meta-heuristic. The other kind of meta-heuristic performs multiple searches at different levels of decision points. This type of meta-heuristic is called hyper-heuristic. It uses different heuristics at various decision points for optimization problems. Meta-heuristics is of various forms depending on the interpretation of what constitutes ‘intelligent’ search. Meta-heuristics is classified into different types based on the features of the basic design of the meta-heuristics. The three basic design elements: (1) the use of adaptive memory, (2) the type of neighborhood exploration used and (3) the number of current solutions carried from one iteration to the next. In this research, we use ‘Tabu Search’ a meta-heuristic method to solve different combinatorial optimization problems.

2.3 Neighborhood

Neighborhood is one of the important aspects of heuristics. By defining the neighborhood function, the solution can be varied. So the neighborhood function is defined according to

the requirements of a given problem. Neighborhood function is defined by using the operations of moving, swapping and replacing.

- According to Mathworld, Neighborhood is defined as to move the point that is in the set to some distance without leaving the set. For a solution neighborhood is defined as the new solutions obtained when a pairwise exchange of any two nodes or replacing the node in the solution with another node is done. So the neighborhood of a feasible solution is always feasible (i.e., does not form any sub-tour). The neighborhood solution obtained by performing swapping or replacing or moving will have a different route from the solution but the number of nodes remains same. The node with the best objective value is selected from the neighborhood to perform the swap or exchange in the heuristic. The concept of neighborhood used in the tabu search is different from that of the local search algorithms. Depending on the type of the move the neighborhood is classified into Constructive neighborhood and destructive neighborhood. If the move applied on the solution results in constructive process then the neighborhood is called constructive neighborhood. Similarly if the move results in destructive then the neighborhood is defined as destructive neighborhood. In tabu search we use dynamic ways to define the neighborhoods. To avoid cycling in the search space we use recency based memory (short term). By using recency based memory we can eliminate the recently visited solutions by making assigning them the status of a tabu. We can also use frequency based memory (long term) to expand the neighborhood and examine the unvisited regions.

2.4 Tabu Search

2.4.1 History of TS

Tabu search was first proposed by Fred W. Glover in an article published in 1986, although it borrowed many ideas suggested before during the sixties, and it was formalized in 1989. The two articles simply entitled 'Tabu Search' [Glover, 1989, Glover, 1990] proposed the majority of tabu search principles which are currently known. This search method was initially introduced to overcome the local optima which are produced during the local search methods or traditional algorithms. The articles by Glover on Tabu search were not well understood in the early nineties and restricted the domain of the principles of the technique. However, tabu search is popular due to the pioneering works by the team of D. de Werra at the Swiss Federal Institute of Technology, Lausanne.

The word Tabu or Taboo comes from Tongan, a language of Polynesia, where it was used by the aborigines of Tonga Island to indicate things that cannot be touched because they are sacred [13]. According to the Webster's dictionary, the word tabu or taboo is defined as 'set apart as charged with dangerous supernatural power and forbidden to profane use or contact…' or 'banned on grounds of morality or taste or as constituting a risk…'. This word is associated with the meaning of guiding the search process to the difficult regions with using restrictions [11].

2.4.2 Problem Definition

For the past, few decades tabu search have been researched and applied to a wide variety of practical optimization problems. The applications are ranging from scheduling to telecommunications and from character recognition to neural networks and are successful in obtaining optimal or near optimal solutions.

According to Fred Glover, the mathematical notation for describing a broad class of problems to be solved by using tabu search is defined as below.

A function f(x) to be optimized (minimizing or maximizing) when subject to $x \epsilon X$, Where f(x) may be linear or nonlinear and the set X summarizes the constraints on the vector of the decision variable x. The constraints may include linear or nonlinear inequalities, and may compel all or some components of x to receive discrete values. For different types of decision problems tabu search can be applied directly without transforming the problem into mathematical formulations [5].

CHAPTER 3

TABU SEARCH FRAMEWORK

In this chapter, we will discuss the basic concepts of tabu search. It also gives information about different strategies for intensification and diversification of the search that are used in tabu search.

3.1 OVERVIEW

Originally Tabu search was proposed by Fred Glover in 1986 to overcome the local optima that are faced by the local search algorithms. Most of the basic important features are proposed by Fred Glover, but most of these features were not used in the initial period of the research of the tabu search. In this chapter, we will discuss the principles of the tabu search that are described by Fred Glover [11] [12] [14]. Tabu search algorithm is used to explore the new areas of the search space. Tabu search uses 'intelligence' to direct the iterative search in a prospective and good direction. So the effectiveness of tabu search in the problem solving, depends on the way how adaptive memory is used and responsive exploration.

3.2 BASIC CONCEPTS

Tabu search algorithm has been improved by many researchers over the past decade to get a preferable solution for a given problem. The main concept of the tabu search algorithm remains the same even though it has been improvised. Tabu search is widely popular because of its use of memory and responsive exploration. These two features are

the most important factors of tabu search in directing the search process of a given problem and finding the solutions apt to the given problem [11].

3.2.1 ADAPTIVE MEMORY

To perform an intelligent search on a problem, the first requirement is to have the data of the past moves of the process. So Tabu search incorporates memory to store the history of the past actions performed at the time of search process. It uses flexible memory structure to store the history. By using memory in Tabu search algorithm to store history faces the challenge regarding the storage space.

3.2.1.1 Explicit and Attributive Memory

The memory which stores the records of complete solutions, especially records consisting of the elite solutions obtained during the search and records the highly attractive but unexplored neighbors of elite solutions. Hence this memory is called *explicit memory*. This memory can be used to define or extend the neighborhood for the tabu search. It intensifies the search process.

The memory which stores the record of information about solution attributes that change in moving from one solution to another. Hence this memory is called *attributive memory*. In general sense, the attributes refer to the values of variables or functions. Attributes can also be combined strategically to obtain other attributes by using hash functions etc. which can be used in the memory. This memory reduces the size of the neighborhood by forbidding some moves in the search.

In tabu search algorithm we use both explicit and attributive memory.

3.2.1.2 Types of memory structures

Depending on the required outcome the tabu search algorithm uses its memory structure.

There are two types of memory structures: *short term memory* and *long term memory*. Each type of memory has its own special strategies and gives different solutions. The use of these memory structures is to modify the neighborhood of the current solution to obtain a new solution. The modified neighborhood is obtained by maintaining a selective history depending on the type of the memory structure used at different states of iteration in the tabu search process.

The memory structures are functioned by referring to four principal dimension based memory: Recency, Frequency, Quality, and Influence.

The *Quality* based memory is used for the ability to differentiate the merit of solutions visited during the search.

The *Influence* based memory is used to impact the choices made during the search. It focuses on quality as well as structure.

Recency based memory is used for keeping the track record of the solution attributes that are recently changed. Short term memory uses the recency based memory.

Frequency based memory is used to keep track of solutions that are very commonly used in the past. The *Recency* and *Frequency* dimensions complement each other.

3.2.2 Responsive Exploration

Tabu search process should use the stored history in an efficient way. To make an intelligent search responsive exploration is an important decision of the search process. Tabu search should use strategic restraints and inducements on the neighborhood of the current solution by using tabu conditions and aspiration levels.

The main idea of this responsive exploration is to direct the search in a more promising way to find a good solution. The search process should be focused on the good regions and good solution features by using intensification process. By using diversification process the search process is extended to exploring the promising new regions.

Strategic oscillation and path relinking process are also features of the responsive exploration. By using these processes on the search space results in new solutions. Path relinking generates new solutions by exploring the neighborhood path of the elite solutions by integrating the intensification and diversification strategies.

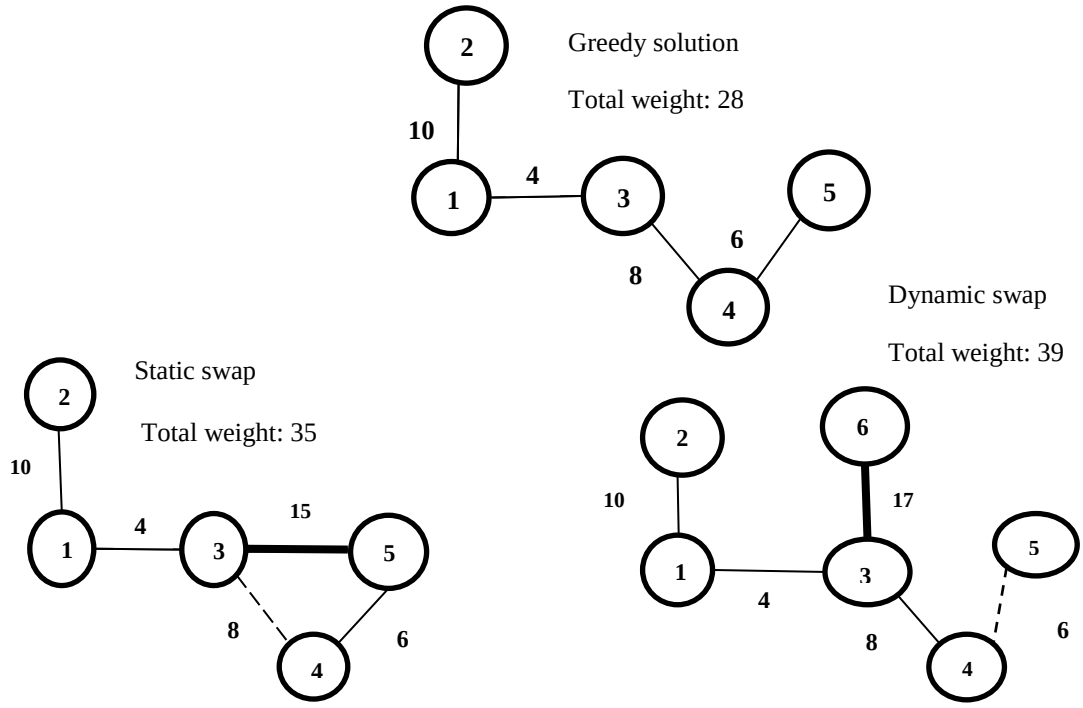
3.3 Move mechanism

In tabu search new solutions are generated by applying move mechanism on the current solution and the neighborhood of the current solution. Move is defined as replacing the edge with a new edge in the neighborhood or swapping the edges. All the moves that are applied to the current solution are stored in the candidate list. Each move can generate a new solution. We need to select a move which gives a better solution or moves the search in a new direction.

Swap move mechanism is used to replace the edge in the current solution with an edge in the neighborhood of the current solution. There are some restrictions in selecting the edge like if the search process is for a tree then the selected edge should not result in any cycles. The selected edge should satisfy some requirements based on the type of the problem. There are two types of edge swaps, static edge swap and dynamic edge swap.

In static edge swap the nodes of the current solution remain same. In dynamic edge swap the edge is swapped by using a new node. The following figure shows the edge swaps.

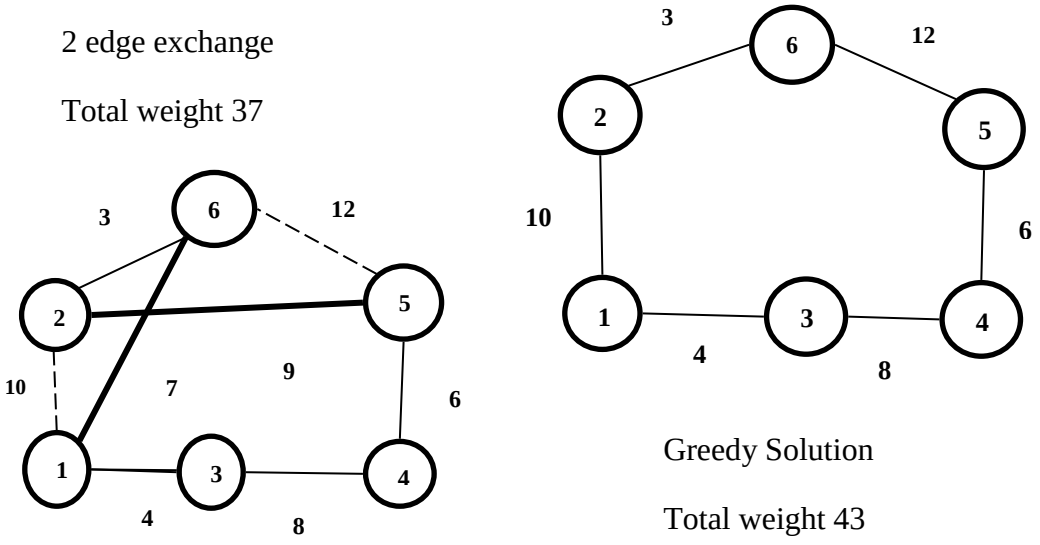
Fig: 1 Swap move mechanism



In Fig. 3.3.1 the dotted lines represent the deleted edge and the heavy line represent the added edge.

Two edge exchange move mechanism is used to connect the two selected edges in the current solution in a different way. Select two edges from the current solution that are to be exchanged and connect the edges in such a way that the obtained solution does not lead to a cycle.

Fig: 2 2-Edge swap move mechanism



The edge exchange swap can be between 3 edges and so on. If it performs 3 edge exchange we need to reconnect them without forming any cycles in the graph.

These are some move mechanisms used in the tabu search. Determine all the move edges and store them in the candidate list. Then select the best move from the candidate list and apply it on the current solution to obtain a new solution. For every step in the search process we should determine all the moves applied on the current solution of the iteration.

3.4 Tabus

How an edge or a node in the solution becomes a restricted element for the next search iteration is defined by defining the tabus. In tabu search Tabus are one of the distinctive elements. Tabus are used to prevent cycling when applying move mechanism on the neighborhood of the current solution. The important point is that in situations like local optima, we need to prevent the search from tracing back its steps to where it came from. This is achieved by declaring the recent moves applied on the solution as tabu. Tabus are stored in a short-term memory of the search called *tabu list* (list of possible moves that cannot be performed on the solution)

In tabu list only a fixed and limited quantity of information is recorded. We can record complete solutions but it requires a lot of storage memory and makes it expensive to check whether a move is tabu or not. We can define the length of the tabu list. The edges in the list can be removed by an FIFO (first come first serve) technique. The most commonly used Tabus record the last few transformations performed on the current solution and prohibit reverse transformations. By applying moves on the initial solution we can obtain a list of Tabus. These moves could be swap operations (as in TSP) or

subtractions and additions (Add/Drop) moves in case of dealing with numeric optimization problems.

3.5 Aspiration criteria

Tabus sometimes may prohibit attractive moves, even when there is no danger of cycling, or they may lead to an overall stagnation of the searching process. So it is necessary to use some conditions to cancel Tabus. These conditions are called *aspiration criteria.* It is a sensitive key factor in tabu search because this defines the flexibility of the tabu search algorithm. The simplest and most commonly used aspiration criterion consists in allowing a move, even if it is tabu, if it results in a solution with a better solution than that of the current best-known solution. Aspiration criteria allow a tabu move to be selected based on certain constraints. For example, the move allows a new global best solution, therefore the move is accepted, and its tabu tenure (tabu length) is renewed. After iteration of the algorithm the tabu tenure is decremented. Only when the tabu tenure of a certain move is 0, can the move be performed and accepted. There are mainly two types of aspirations: move aspiration and attribute aspiration. Attribute aspiration criteria are used to revoke the active tabu status of the attributes. Move aspiration criteria are used to revoke the solution's tabu classification.

By default the aspiration criterion is to select least tabu one if all the available moves are in tabu list. One category of aspiration criteria is if the direction suggested by using an attribute improves the solution then revoke the tabu status of that attribute. Aspiration criteria can be based on influence of the solution such as the measure of the degree of change in solution structure, quality or feasibility.

3.6 Intensification

In order to find best solutions we should search more thoroughly the parts of the search space that seem promising. For more effective search we use additional components of tabu search like intensification and diversification strategies. Intensification is a form of exploitation. Intensification strategy is based on improving the choice rules. Intensification is based on intermediate-term memory, such as recency memory, in which more move combinations are applied on the initial solutions or best solutions considered from the traditional algorithms without interruption. It generates neighbors by either using grafting or by using evaluation strategies of good solution. Intensification is used in many tabu search implementations, but it is not always necessary because in many situations the normal search process performed is already thorough enough. So there is no need to examine more carefully on the parts of the search space that have already been visited.

Even with the use of tabus, the tabu search is still tending to be local, i.e., they tend to spend most, if not all, of their time in a limited portion of the search space. The negative consequence of this fact is that, although we achieve good solutions, but we fail to explore the other interesting parts of the search space and thus end up with solutions that are still pretty far from the optimal ones. Diversification strategy is used to address this problem by forcing the search into previously unexplored areas of the search space.

3.7 Diversification

Diversification strategy is based on long-term memory of the search, such as frequency memory. Diversification is a form of exploration. Diversification means extending the search to a new direction. It records the total number of iterations from the beginning of the search that are produced in the current solution and also the moves that have been performed on the initial solution. The diversification strategy is the most critical issue in the design of tabu search heuristic. It should be handled with extreme care from the initial step and it should be revised if the obtained results are not up to expectations. By using diversification strategies the search process is used in directing the search to new regions of the search space. In tabu search, intensification and diversification are not entirely separated. If the algorithm is restricted to a particular search space and is difficult to guide the search to different search regions then we mostly restart tabu search algorithm with different initial solution. Instead of randomly choosing this initial solution for restarting, it is efficient to use systematic diversification.

Some mechanisms used to implement intensification and diversification strategies are: path relinking, strategic oscillation, reinforcement by restriction, solutions evaluated but not visited, creating new attributes etc.

CHAPTER 4

TABU SEARCH ALGORITHM

This chapter gives the insight of the general steps of implementing the basic level of tabu search algorithm. The generalized algorithm in this chapter is obtained by surveying papers published by Fred Glover and also with the reference of the book written by Fred Glover and Manuel Laguna [11] [12] [13] [14].

4.1 OVERVIEW

Tabu search is a local search based algorithm with three primary ideas. The first idea is the use of flexible memory structures to search and evaluate the information of the past moves performed on the solution. The second idea is to control the moves to be applied on the solution at the time of search process. The third primary idea is to use memory functions of different time spans like short term memory and long term and can also perform different strategies on intensifying and diversifying the search. The following chapter shows how these ideas form the basic structure of the tabu search algorithm.

4.2 Find the trial/initial solution

To perform a Meta heuristic Tabu search algorithm first we need to have an initial or trial solution from one of the traditional algorithms or other heuristic search methods or it can be randomly generated. While using tabu search algorithm for NP-hard or NP-complete problems, selecting the initial feasible solution is one of the important step for obtaining a good solution. Tabu search algorithm depends on the selection of the initial solution. The tabu search moves in the direction of the selected original solution since we consider the neighborhood of the initial solution to continue the search. This solution can be from

other meta heuristic algorithms like simulated annealing, genetic algorithm or even from tabu search.

4.3 Creating a candidate list

After obtaining the initial solution find all the possible moves from the neighborhood of the current solution that can be applied on the initial solution. Before selecting the moves define the neighborhood of the solution.

In this implementation we are considering the tabu search short-term memory component. Each move applied on the current solution results in a new solution. For every iterative step generate a candidate list and select the best possible move. The best possible move is selected based on two types. First, select the move if it reduces the overall weight or length of the current solution. Secondly, if there is no move that can reduce the overall weight of the current solution then select the best move in which the overall weight is slightly greater than the current solution. By doing this the search process will be extended to different regions.

The moves created in the tabu search algorithm are an iterative process. This iterative process consists of the following steps.

1. Identify the set of moves applied for the current solution. The selected move should satisfy the following two criteria.

A. The selected edges are either not on the tabu list or they are able to override the tabu status.

B. The selected edges which would produce the largest decrease in the tour length are selected as the best. If no improving moves exists, then select the one which would produce the smallest increase in the tour length.

2. Perform the move mechanism with the edge identified from the previous step. This will result in a new slightly different tour than the previous tour.
3. Update the tabu list and aspiration list to save the information of the edges that are applied in the previous step.
4. If this tour length is better than the previous best tour length found in the search until now then update the best tour information.

The above process should be repeated until the stopping criterion is satisfied. If it satisfies, the search is terminated and the information about the best tour found by the search will be one of the best solutions for the problem.

The basic tabu search algorithm can be described in the following way.

algorithm tabu search
begin
tabu_list:= [];
S:= initial solution;
S*:= S;
R**epeat**
find the best admissible solution S_1 belongs to Neighborhood of S
if f(S_1) > f(S*) **then** S*:= S_1;
S:= S_1;
Update Tabu list tabu_list;
Until stopping criterion;
End;

4.3.1 Tabu list and Aspiration levels

To prevent the reverse of an exchange in a short period, we use tabu list. In tabu list, we store the selected attributes of the edges performed in the algorithm. The goal of using this list is to prevent the situation where the search identifies the same sequence of tours over and over again, this situation is called cycling. The tabu list should be implemented

in a way that, the search should not be overly restricted in its ability to look for better solutions. The two factors that are used in relieving the status of the edge in the tabu list are finite length of the tabu list and the aspiration criterion.

The tabu list length can be defined according to one's requirement of releasing the status of the edges from tabu. Initial number of exchanges is stored in the tabu list. Once the exchanged edges in the list are equal to the length of the tabu list, the edges in the list removed by a method called FIFO (first come first serve).The exchanged edges will replace the oldest edges in the tabu list. In this way the oldest edges which are being replaced in the tabu list loose the status as tabu. The occurring of a cycle is greatest, right after the exchange is made and decreases when more and more exchanges are made. So removing the edge from a tabu list after more exchanges are made leads to less occurring of the cycles. The tabu list corresponds to the sufficient number of exchanges that will prevent cycling but will not excessively reduce the pool of candidate exchange. The finite length of the tabu list makes it to act as a short term memory.

By using aspiration criterion the tabu status of the edge in the tabu list is overridden. The tabu status of the nodes or edges is not fixed, it can be overruled if certain conditions ae met, which are expressed in the form of aspiration levels. So if the move satisfies the aspiration level then, it can be admissible even if the move is in the tabu list. These criteria are designed to override the tabu status if a move is sufficiently limiting to the goal of preventing the solution process from cycling. Generally used or most simple form of an aspiration-level check is to permit tabu status to be overridden if the solution produced would be better than the current best solution. Another approach is to define an

aspiration level. If the search is moving in a new and promising direction, then this criterion will allow the tabu restriction to be relaxed.

4.3.2 Tabudrop and Tabuadd

One of the important aspects is to determine which edge or node is to be placed in the tabu list. The edges to be placed in the tabu list after an exchange is determined by the two parameters namely tabuadd and tabudrop. We can have three possible combinations of edges from an exchange to be classified as tabu: added edges only, dropped edges only, both added and dropped edges. We can use any combination to keep an edge in the tabu list. Depending on the problem requirement we can use one of these combinations.

If tabudrop is assigned 1, then the added edges of the 2-edge exchange are placed on a tabu list to prevent them from being dropped from the tour by a subsequent exchange. If tabudrop is assigned 0, then the added edges are not placed on a tabu list.

If tabuadd is assigned 1, then the dropped edges of the 2-edge exchange are placed on a tabu list to prevent them from being added back into the tour by another exchange. If tabuadd is assigned 0, then the dropped edges of the exchange are not placed on a tabu list.

4.4 Stopping criterion

Stopping criterion is used to determine the end of the tabu search. The stopping criterion can be the number of specified iterations to occur at the time of tabu search. Initially we can define for how many iterations the search process should repeat. It counts the total

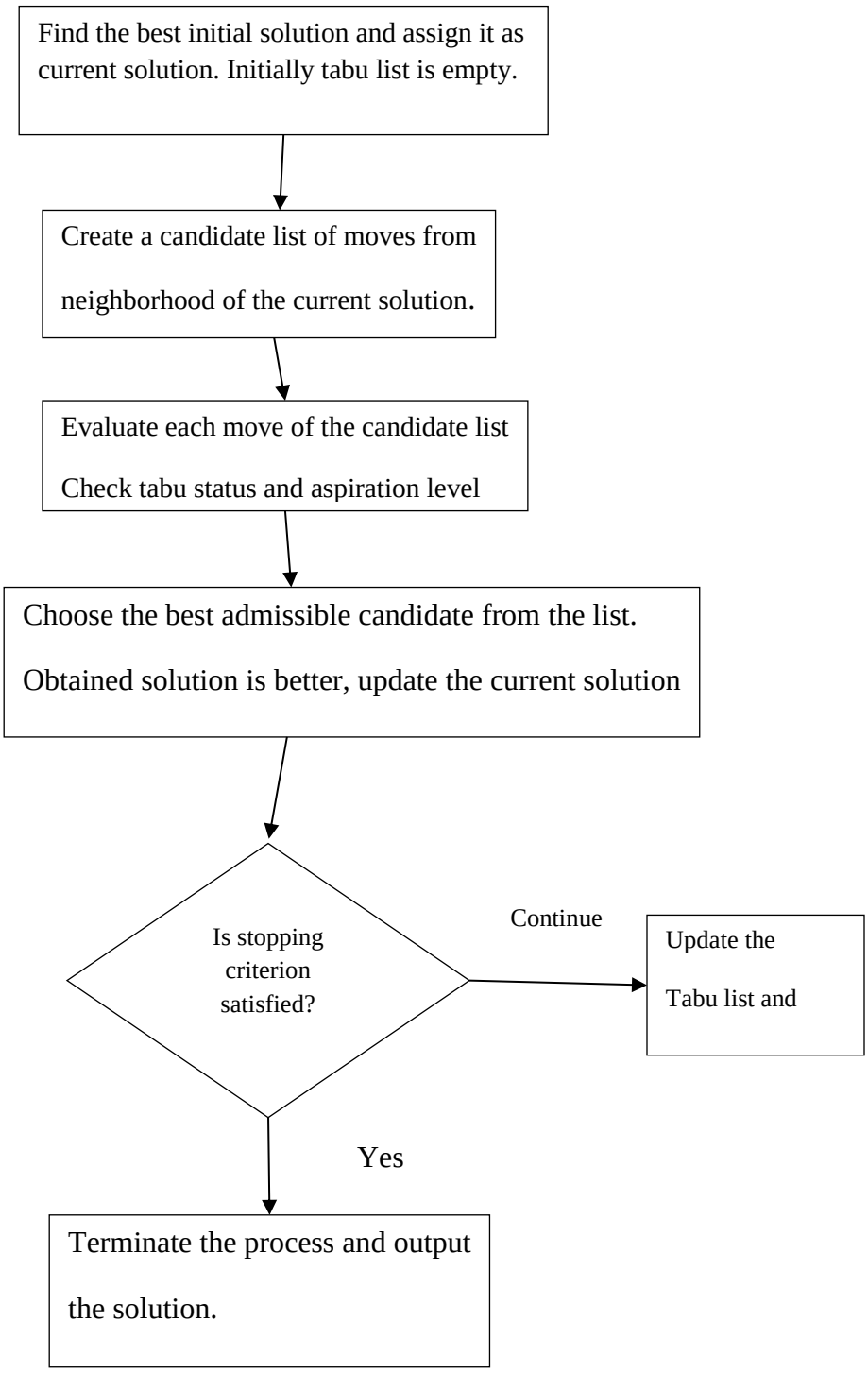
number of iterations occurred and if it is equal to the defined number of iterations then the tabu search process terminates and outputs the best solution until now.

The stopping criterion can be a fixed number of iterations occurred after finding the best solution. Let the process repeat for n number of iterations after the best solution found from the process. The stopping criterion can be defined according to the requirements of the problem and the type of solution required.

We can describe the above tabu search algorithm with a flow chart. The following flow chart is the generalized version of the tabu search algorithm. In the flow chart, at move mechanism for iteration of the tabu search algorithm there will be a list of candidate moves. To apply a move, initially we need to consider all the possible moves on the current solution and create a candidate list. From the candidate list we will consider the best move and finally that move will be applied on the current solution. If the obtained solution is better than the current solution then update the best solution variable to the present solution performed. For the next iteration the candidate list should be updated with the next set of moves possible on the current solution.

Flow chart of the tabu search algorithm can be described as follows.

Fig: 3 Flow chart for tabu search algorithm



CHAPTER 5

APPLICATIONS OF TABU SEARCH

5.1 OVERVIEW

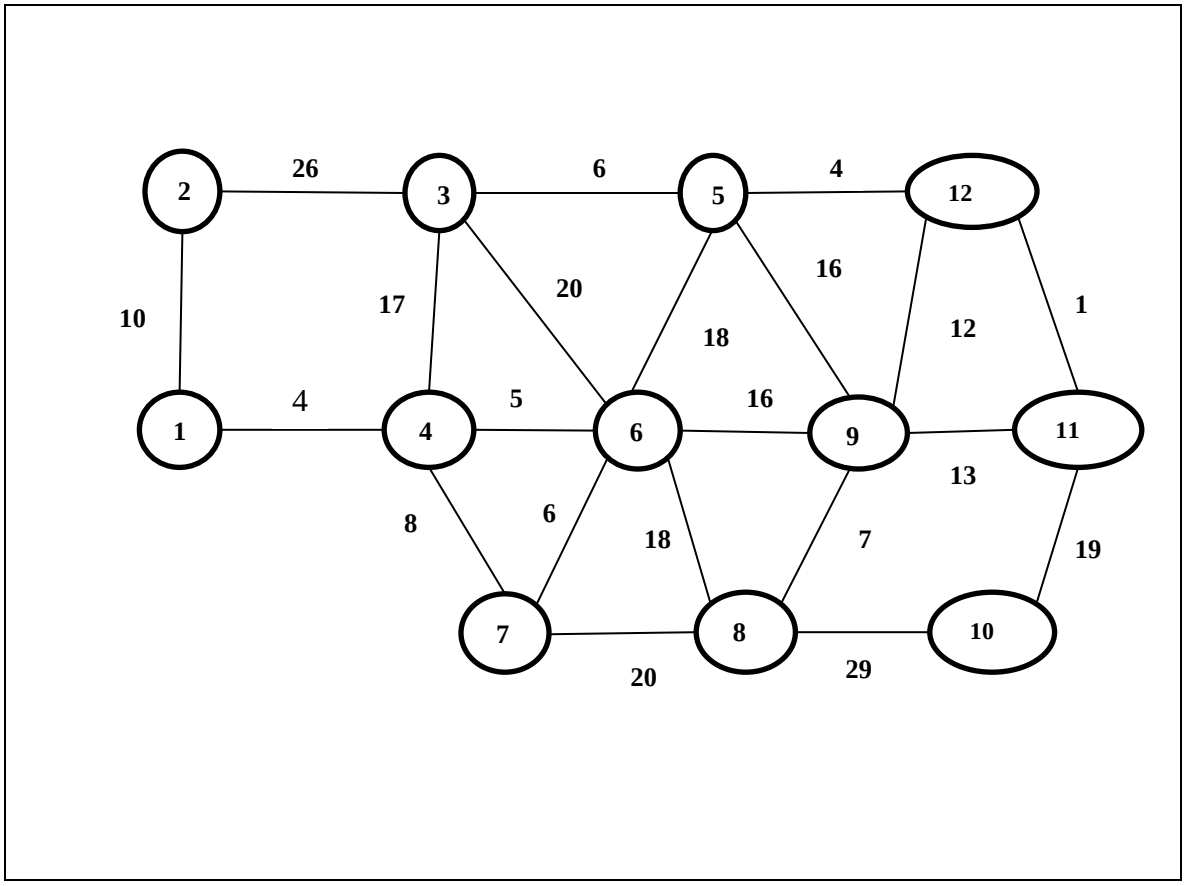
In this chapter we will perform tabu search meta heuristic algorithm to three different problems and the improvements and back drops of implementing tabu search on these examples. The examples that are explained in this chapter are one of the most studied and experimented problems of the combinatorial optimization problems.

5.2 Minimum K-Tree Problem Example

5.2.1 Problem Definition

A tree which consists of k edges in a graph with minimum sum of the weights of these edges is called as minimum K-tree problem. A tree is defined as a set of edges that contains no cycles. A cycle is defined as set of edges connected from one node to different nodes and end with a node in the previously connected nodes. Consider the following figure Fig.5.2.1, where nodes are shown as numbered circles and edges are shown as lines that join pairs of nodes. Edge weights are shown as numbers on the edge lines.

Fig: 4 Weighted undirected graph



5.2.2 Tabu search

5.2.2.1 Initial Solution

Implement greedy algorithm on the given graph to find the initial solution. Greedy algorithm starts by choosing an edge (i, j) with the smallest weight in the graph, where i and j are the indexes of the nodes that are the endpoints of the edge. Then the next smallest edge is selected which is connected with one of the previous edges and continue the process similarly but select it in a manner that at any particular point there should not be any cycles in the graph. Below is the Greedy construction table for k= 4 edges. After

performing greedy algorithm on the above figure for 4 edges we got the initial solution. So in this particular case the initial solution has a total weight of 28.

Table 1 Greedy construction

Step	Candidates	Selection	Total Weight
1	(1,4)	(1,4)	4
2	(1,2), (4,3), (4,6), (4,7)	(4,7)	12
3	(1,2), (4,3), (4,6), (7,6), (7,8)	(7,6)	18
4	(1,2), (4,3), (4,6), (7,8), (6,8)	(1,2)	28

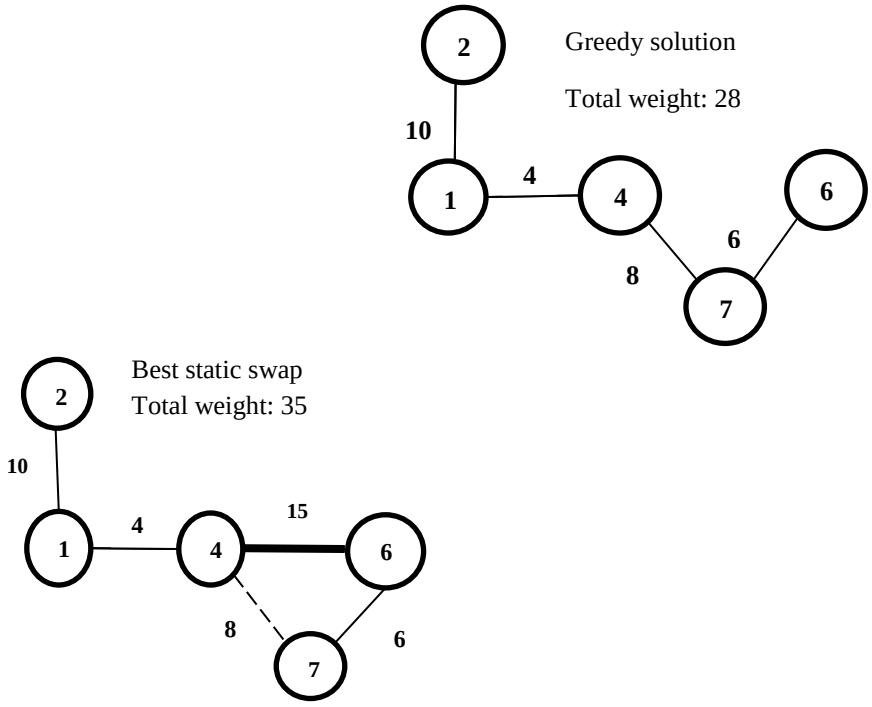
5.2.2.2 Candidate list of moves

Apply move mechanism to the initial solution. For this example, move mechanism can be defined by edge-swapping. Edge swapping is defined as replacing the current edge in the solution with the edge in the neighborhood in resulting a new solution with the same number of edges. Each move applied on the initial solution would generate a new solution. Create a list to store all the solutions generated by applying the move mechanism.

Perform swap move mechanism on the sub graph which is obtained from greedy construction in 5.2.2.1. In this mechanism, replace an edge in the current tree (i.e. the sub graph we obtained above) by another edge in the neighborhood of the graph in such a way that the resulting sub graph is also a tree. There are two types of edge swaps, one the current nodes of the tree are unchanged (Static) and the other that results in replacing a node of the tree by a new node (Dynamic). The following figure shows the first swap

move mechanism applied on an instance of the graph. In the figure below the heavy line is the added edge and dotted line is the deleted edge.

Fig: 5 Move mechanism applied on initial solution



After doing swap move on the initial solution the best solution obtained i.e. the total weight is increased. Here the best solution is taken as the objective function value. Since the total weight is increased in the current situation, this move is abandoned by the approach and thus leads to the tabu search process.

The next step in tabu mechanism is to choose the key attributes that will be used for the tabu classifications. This is one of the key points of the tabu mechanism; considering the type of the problem and which type of solution is to be obtained we can assign the key

attributes to settle on that particular design. In this example since the moves are defined by adding and deleting edges, these edges can be used as the attributes. After identifying the key attributes now we need to choose the tabu classifications. Tabu classifications are not specifically symmetric, so the tabu structure can be designed by treating added and dropped edges differently.

In the above figure edge (4,7) is deleted and edge (4,6) is added. So let us assign the tabu status for both of these edges. Since tabu search is very flexible we have different possibilities. One of the possibilities is to make these two edges as tabu-active for the same number of iterations. The tabu-active status has different meanings depending on whether the edge is added or dropped. For the added edge, the tabu-active means that this edge is not to be dropped from the current tree for the specified number iterations during its tabu tenure. The number of iterations an edge can take is called the tabu tenure of the edge. For the deleted edge, the tabu active means the edge is not allowed to be added or included in the current solution during its tabu tenure. Since there are many edges outside the graph it is efficient to use dropped edges so that the search can be expanded to outside the graph. So we will keep the recently dropped edge tabu-active for a longer period of time than a recently added edge. If the tabu solution after moves gives a better solution than the initial solution then the current solution is considered and the initial solution is overridden. This is called improved-best aspiration criterion.

Table 2 Iterations of a first level TS procedure

Iteration	Tabu-active net tenure		Add	Drop	Move value	Weight
	1	2				
1			(4,6)	(4,7)	7	35
2	(4,6)	(4,7)	(6,9)	(6,7)	10	45
3	(6,9), (4,7)	(6,7)	(8,9)	(1,2)	-3	42
4	(8,9), (6,7)	(1,2)	(9,12)	(1,4)	8	50
5	(9,12), (1,2)	(1,4)	(11,12)	(4,6)	-14	36
6	(11,12),(1,4)	(4,6)	(5,12)	(6,9)	-12	24
7	(5,12), (4,6)	(6,9)	(3,5)	(8,9)	-1	23

By surveying many papers on solving K-tree problems with tabu search [5] [15] [16], it is one of the better solutions to get better solutions.

We can improve this basic tabu search algorithm by making small changes and get a better solution. For example, by using the restarting procedure we can search the nodes of the graph that are not examined before. By doing this we might get better solutions than the above obtained results. In this method, select a new edge which is not in the sub graph and create a new solution. Restrict the edges in the sub graph from not using them in the new solution. By doing this we can search a new region and find solutions in the graph.

5.3 TRAVELLING SALESMAN PROBLEM

The travelling salesman problem (TSP) is a classic problem in combinatorial optimization research, because it is relatively easy to describe but extremely difficult to find an optimal solution. Travelling salesman problem is an NP-hard problem, so it is hard to solve all TSP instances to optimality within a reasonable execution time. As we

discussed in the above chapter the method to perform tabu search algorithm, we will use the method on TSP.

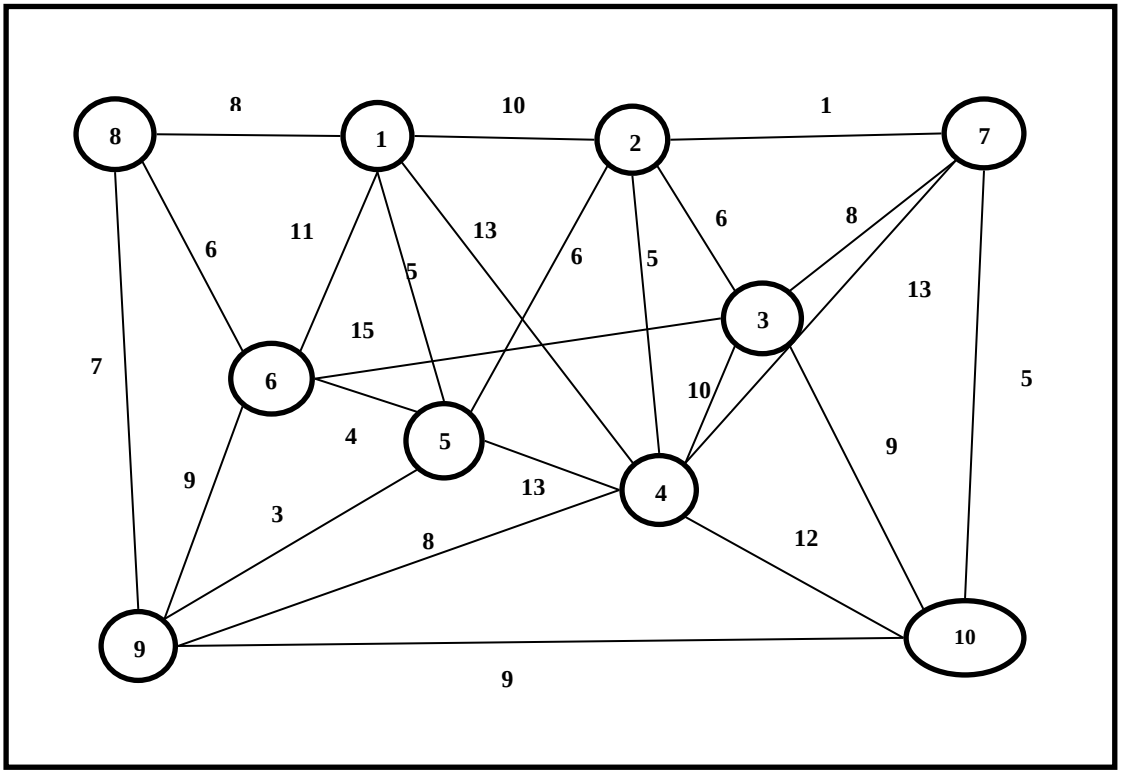
5.3.1 PROBLEM DEFINITION

Travelling salesman problem is defined as finding a minimum cost route in an undirected graph from an initial starting point and covering all the points in the graph exactly once and coming to a stop at the started initial point.

Travelling salesman problems may be symmetric or asymmetric. In symmetric problems the cost of an edge is independent of the direction of travel (i.e., the cost of travelling from city A to city B is always same as from travelling from city B to city A). In asymmetric problems the cost of an edge may be dependent on the direction of travel (i.e., the cost of travelling from city A to city B may be different from the cost of travelling from city B to city A). In this example we will discuss with the problems concerning only symmetric TSP problems.

Below in Fig.6 it shows a graph of travelling salesman problem with 10 nodes. It is a symmetric graph of travelling salesman problem. The numbers on the lines are the weights of the edges that are connected from one node to another.

Fig: 6 Symmetric Graph of TSP

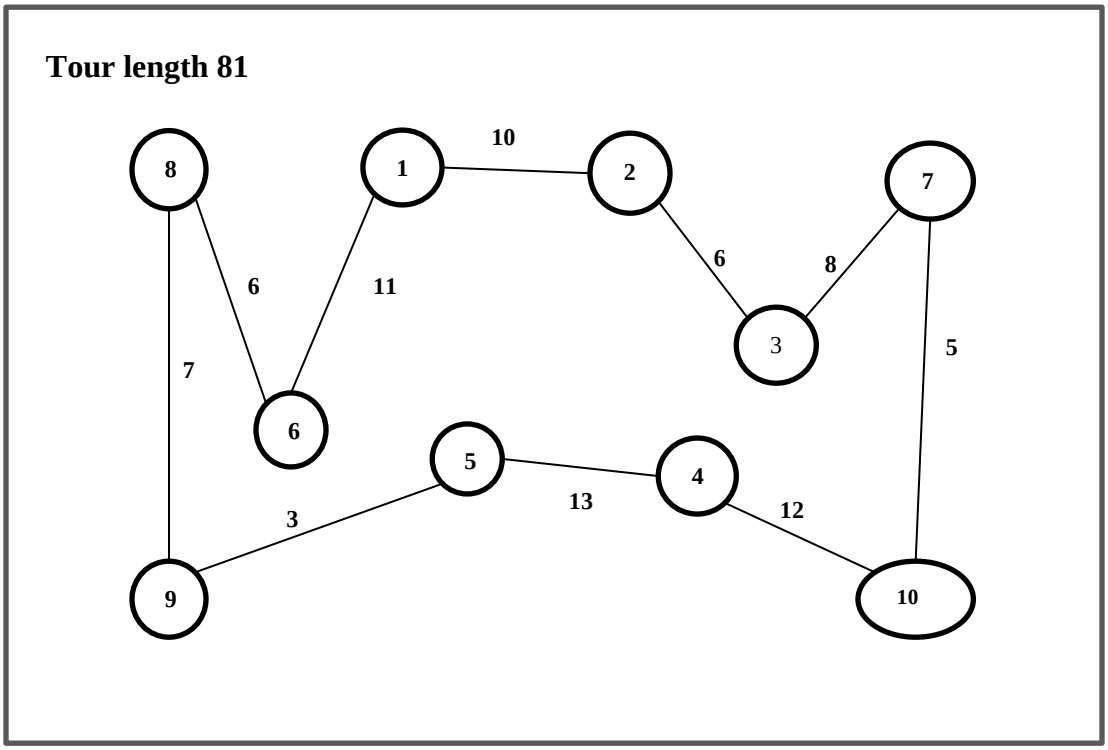


5.3.2 Initial Solution

In this graph we will consider the initial/starting point as node 1. So for the travelling salesman problem we should travel from node1 through all the nodes exactly once in the graph and end with the node1. Consider the longest path travelled in the graph from node 1 so that tabu search can be explained for this example. To get the initial solution we can consider any local search algorithm. Here in this example we used the greedy algorithm to find the longest travelled path. In general if K-OPT algorithm is used to get the initial solution the effect of finding the optimum solution for travelling salesman problem increases.

The trial or partial solution obtained from greedy algorithm is

Fig. 7 Initial solution for Symmetric TSP



The travelling distance of the trial solution is **81**. So we will consider this solution as initial solution or initial input to the tabu search algorithm.

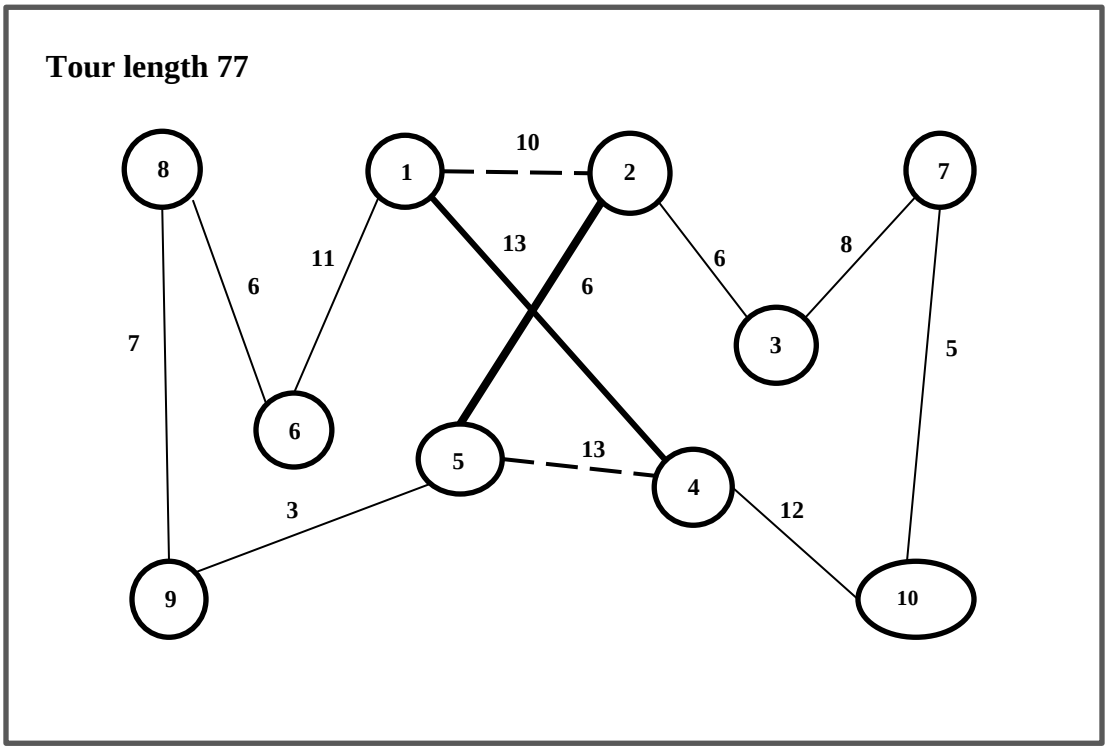
5.3.3 Tabu search algorithm

5.3.3.1 2-edge swap move mechanism

Initially when the tabu search algorithm begins, the tabu list is empty so the selected 2-edge exchange will not be restricted. There won't be any restrictions on selecting the best candidate. It performs an exhaustive evaluation of all 2-edge exchange candidates on each iteration of the search. The tour is rebuilt by replacing the dropped edges with the

two new added edges. In the example the best candidates for exchange are (1,2) and (4,5). These two edges are deleted and (1,4) and (2,5) are added. The new tour length after dropping and adding edges is **77.**

Fig: 8 Graphical Representation of TS iteration 1



By performing the 2 edge move mechanism on the current solution we have obtained a better solution than the initial solution. So the best solution variable will be updated with this solution.

Fig: 9 TS iteration 2

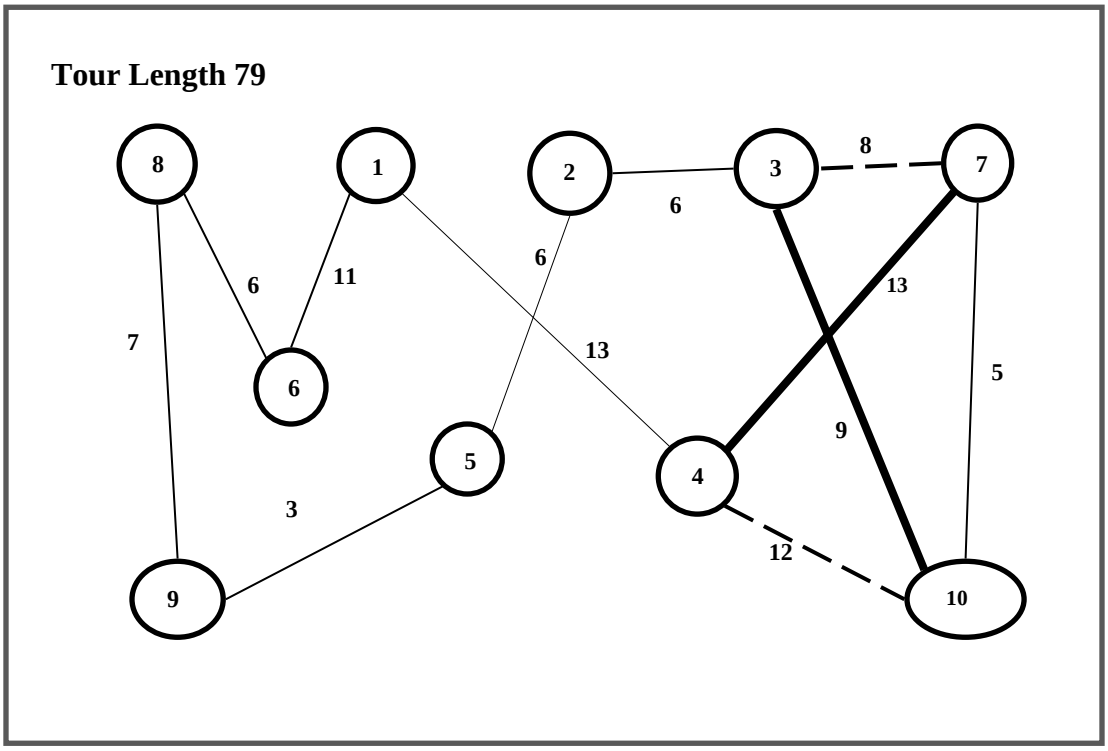


Fig: 10 TS iteration 3

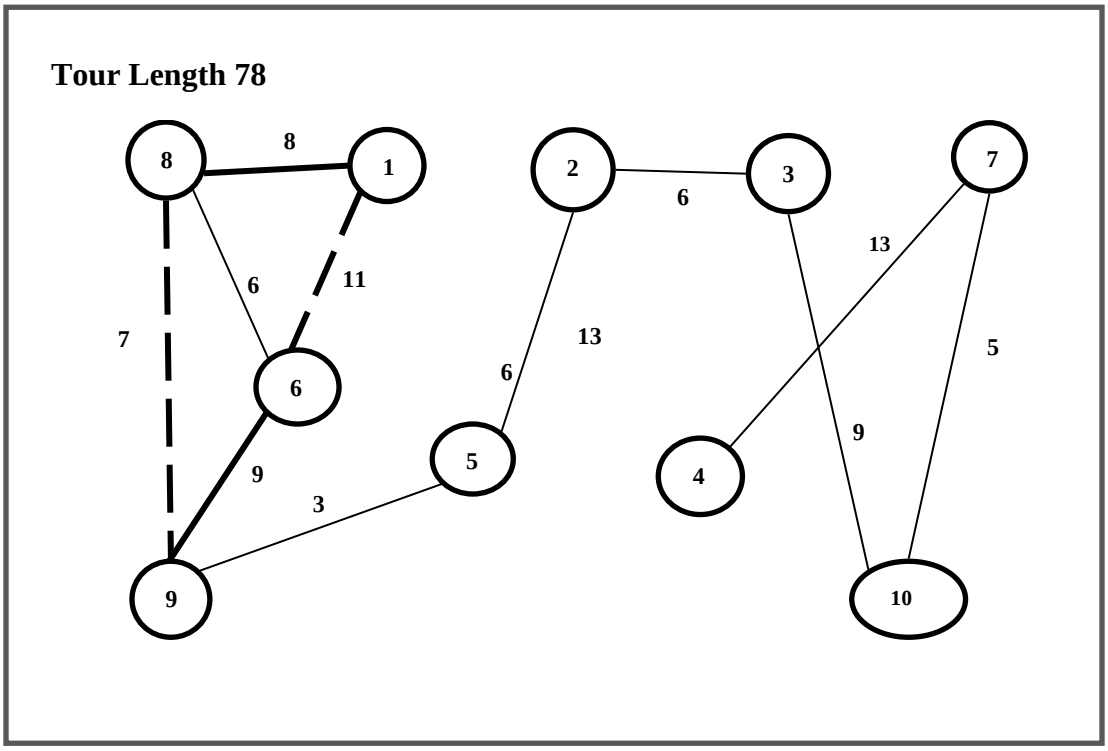


Table 3 Iterations of first level TS procedure for TSP

Iteration	Tabu-Active Net Tenure		Add	Drop	Move Value	Tour Length
	1	2				
1			(1,4) (2,5)	(1,2) (4,5)	-4	77
2	(1,4) 2,5)	(1,2) (4,5)	(3,10) (4,7)	(3,7) (4,10)	2	79
3	(1,2)(4,5) (3,10) (4,7)	(3,7) (4,10)	(1,8) (6,9)	(1,6) (8,9)	-1	78

Repeat the iterations until the stopping criterion is satisfied. One of the stopping criterions can be a fixed number of iterations specified before the process is started. There aren't any back drops by using this method. But we need to identify a good number to get a better solution. Until now the best solution obtained for the graph of travelling salesman problem that we have considered is **77**.

5.3.4 Outcomes for TSP

Many researches have been done until now on solving travelling salesman problem with tabu search heuristic. With small modifications in the basic tabu search algorithm can produce different outcomes for the travelling salesman problem. By surveying many papers [13] [17] [18] [20], using tabu search gives some of the best results for travelling salesman problem. Knox implemented tabu search on TSP in 1989. The quality of the best solution obtained by tabu search depends on the length of the each search and number of searches. For large test problems of TSP, the tabu search gives effective solutions. Many new methods have been applied on TSP while using tabu search, some of them are angle based tabu search, parallel adaptive tabu search, multi point tabu search [23] [24] [25]. The speed of attaining an optimal solution depends on the initial solution. If the initial solution is far away from the optimal solution then the computational time of tabu search increases. So by introducing a multiple structure we can improve this problem. Consider multiple initial solutions and perform the tabu search, after the iteration compare the solutions and assign the best solution obtained by them as the current solution and continue with the tabu search process. By considering the small changes in tabu search algorithm we can improve the performance of tabu search on

travelling salesman problem. Genetic algorithms and simulated annealing can be used in the tabu search for better outcome.

Many papers were written on tabu search with varying factors like tabu tenure, memory structures, aspiration criteria. For example, I surveyed about 30 papers in which 16 papers deal with fixed tabu tenures and the rest of the papers deal with tabu tenures that vary with the iterations of the tabu search or the instance size. Some researchers tried to find optimal tabu tenures for different size of problems but not able to come up with any optimal tabu tenure.

I have researched around 10 papers where long term memory is used to diversify the search to new regions. These papers provide long term memory structures to achieve diversification strategy.

5.4 JOB SHOP SCHEDULING PROBLEM

Job shop scheduling is one of the more difficult combinatorial optimization problems. It has been studied for a long time and is known as NP-hard problem. It is an extremely hard problem because it requires a large search space and it should maintain a precedence order for the machines.

5.4.1 PROBLEM DEFINITION

Scheduling is defined as allocating one or more machines for a job at one or more time intervals [26]. The classical Job shop scheduling problem consists of *n* jobs which are to be processed on *m* machines. Each job consists of a sequence of different operations. Each operation should be performed on a given machine without any interruptions. Job

shop scheduling problem can be described using three field classification introduced in Graham et al. [27] as follows.

A set M of *m* machines and a set J of *n* jobs, where *i*th job consists of a chain of m_i operations from set O = {1,…., N}, with N = $\sum_{k=1}^{n} m_k$. These operations had to be processed on machine for t_i consecutive time instants. The problem is to assign the operations to time intervals by considering the following constraints:

- There is no job precedence
- Precedence of the order of the operations is considered.
- No two jobs are processed on the same machine at same time.
- The makespan of all the operations should be minimized.

Consider the following instance with three jobs and four machines to show the work of basic tabu search algorithm on solving the job shop scheduling problem.

Table 4 Jobs 3 Machines 3

Jobs	**Machine Sequence**	**Processing Times**
1	1, 3, 4, 2	p_{11}= 3, p_{13}=8, p_{14}=9, p_{12}=4
2	2, 1, 3	p_{22}=6, p_{21}=3, p_{23}=5
3	3, 4, 2, 1	p_{33}=8, p_{34}=5, p_{32}=7, p_{31}=1

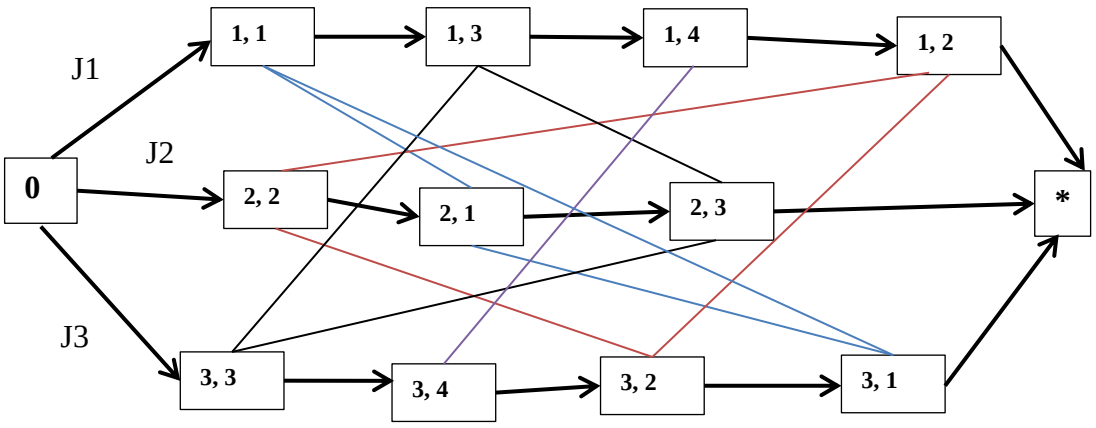
To represent the job shop scheduling problem, Disjunctive graph is one of the most popular models. The following graph is the disjunctive graph representation of the above job shop problem instance. The conjunctive edges in the graph reflect the precedence constraints of the operations of a job. These edges should not be altered in any process of the tabu search.

By changing the direction of the disjunctive arcs in the following graph by considering the constraints we can minimize the makespan of the operations. The constraints to be considered are:

- The resulting graph should be acyclic
- The disjunctive undirected edges are converted into directed conjunctive edges.
- The resulting new path in the graph should consist of minimal processing time.

By aligning the disjunctive lines in the following graph provides the solution to the job shop scheduling problem.

Fig. 11 Disjunctive graph



Representation:

Conjunctive arc: ⟶

Disjunctive arc:

5.4.2 Initial solution

The initial solution for job shop scheduling can be obtained from branch and bound algorithm. Find the feasible initial solution to start the tabu search algorithm. By selecting the best admissible initial solution results in a good optimal solution. We use gantt chart representation to get the initial solution to the given problem. Gantt chart is a simple graphical representation and it does not have any rules in assigning the operation to a machine.

Fig 12 Gantt chart

M1	J1 (10)					J2(3)		J3(1)				
M2	J2 (6)				J3 (7)						J1 (4)	
M3		J3(5)				J1(8)			J2(9)			
M4	J3(2)								J1 (6)			

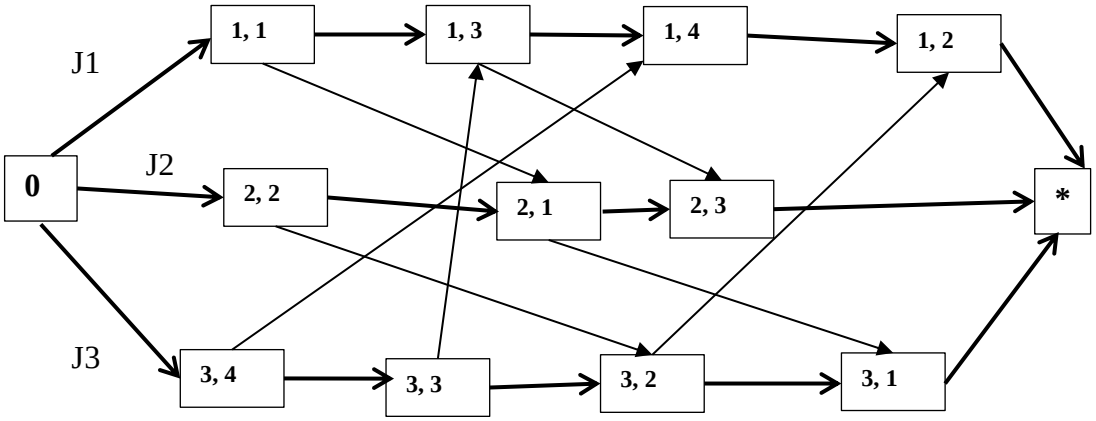
Total makespan of the above gantt chart is 28.

The above graph can be represented in a graph. The graph is machine oriented.

In the following graph, the machines are connected according to the sequence of the operations performed on the machines.

This machine sequence will be the initial solution for the tabu search. The disjunctive edges in the figure 11 are replaced with conjunctive edges of the machine sequence.

Fig: 13 Initial Solution for JSP



5.4.3 Tabu search algorithm

Next step is to define the neighborhood and collect all the moves applied on the current solution. The move mechanism can be swap move mechanism and also the direction of the edge can be reversed. The constraints of this graph are the sequence of the operations of a job should not be altered. Only the disjunctive edges represented in the figure 5.4.1 can be changed or altered. The other constraint is that the disjunctive edges should be particular to that machine. Only the edges of similar machines can be swapped or the direction of an edge can be reversed. The neighborhood defined in this algorithm is opt-connected. Connectivity is the desired property of tabu search. By using connected neighborhood we can achieve better solutions. Define the aspiration criteria for the problem, so that the tabu status of a move can be cancelled. Now select an edge and reverse its direction to see the change of the solution.

Consider edge between job 2 and job 3 for machine 2, change the direction of this edge. By doing this the makespan of problem is increased. The total makespan for the graph is 32.

By performing the move mechanism on the initial solution the obtained makespan is increased. But still we will consider this solution and perform the move mechanism. Later if we perform move mechanism on this solution the makespan can be reduced than the initial solution. By performing this move mechanism we won't be stuck in the local optima and can explore the search region more diligently

Fig: 14 Graph representation of iteration 1

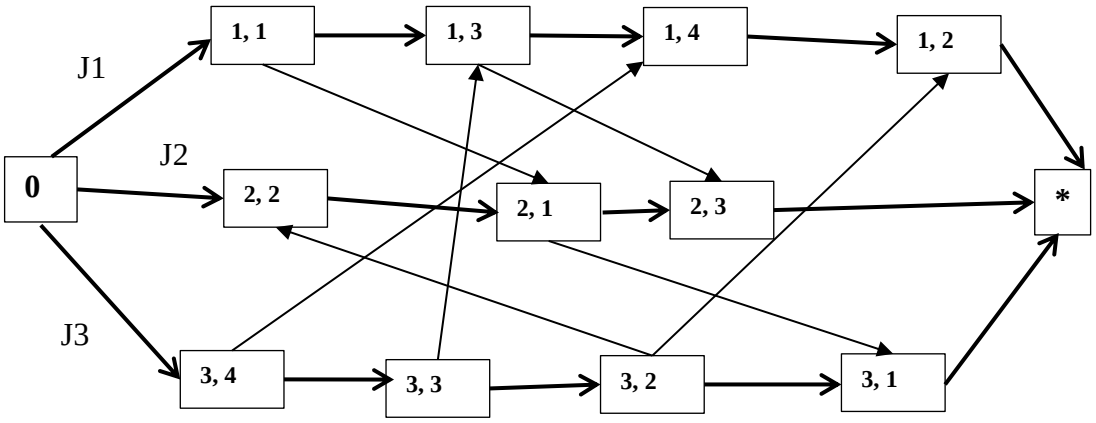


Fig: 15 Gantt chart for iteration 1

M1	J1 (10)					J3(1)			J2(3)			
M2				J3 (7)				J2(6)			J1(4)	
M3		J3(5)			J1 (8)							J2(9)
M4	J3 (2)							J1 (6)				

We can get different solutions by applying moves on the current solution. If there is a minimum makespan than the current solution, then update the current solution to the solution obtained.

5.4.4 Observations of Tabu search on JSP

Job shop scheduling problem has excellent practical applications. This problem has been researched for a long time. Many papers were published on Job shop scheduling with tabu search algorithm. In 1989 Eck used tabu search for solving job shop scheduling problem.

By researching papers on job shop scheduling problem [31] [32], it is noticed that with the increase of the size of the problem, the neighborhood for the solution increases rapidly. So to find the admissible moves in the neighborhood becomes difficult and also computational time increases. Researches are focusing on reducing the size of the neighborhood without decreasing the quality of the search (Jain Et. Al, 2000).

By combining tabu search with simulated annealing or genetic algorithms a hybrid algorithm is produced which gives better solutions for the NP complete problems.

CHAPTER 6

CONCLUSION AND FUTUREWORK

Tabu search algorithm is applicable for problems in different fields of the technology, Communication. This algorithm is able to provide better solutions to many combinatorial optimization problems. The efficiency of the solutions obtained by tabu search is mostly depended on the initial solution and neighborhood structure. The future scope of this algorithm is to find an algorithm which is less dependable o initial solution and decrease the size of the neighborhood with the increase of the problem size. Even with the first level of tabu search it provided better solutions to some of the bench mark problems. By combining the intensification and diversification strategies the search is more directed to the different regions of the search space there by providing much better solutions. With the tabu constraints and aspiration levels on the search leads to effective solutions. There are so many areas in which research can be developed like strategies for intensification and diversification.

In present life, the applications of tabu search are increasing rapidly. This alone suggests the potential approach of the algorithm and its principles. It is an art to develop the tabu search algorithm for particular type of problems.

BIBLIOGRAPHY

[1] Zbigniew Michalewicz, David B. Fogel, *How to Solve It: Modern Heuristics*, Second Edition, Chapter 1, 2, 5, 6, Springer, 2004.

[2] Reeves, Colin R, *Modern Heuristic Techniques for Combinatorial Problems*, 1993.

[3] Graham, Ronald L., Grötschel, Martin, Lovász, László, *Handbook of Combinatorics*, Chapter 28, 1995.

[4] J. Dreo, A. Petrowski, P. Siarry, E. Taillard, *Metaheuristics for Hard Optimization*, Methods and Case Studies, Book coordinated by Patrick Siarry, Springer.

[5] Glover, Fred and Laguna, Manuel, *Tabu Search,* 1997, Kluwer Academic Publishers, Boston. This is the first comprehensive book on tabu search.

[6] Bernhard Korte, Jens Vygen, *Combinatorial Optimization, Theory and Algorithms*, Fifth Edition, Chapter 2, 6, 15, 21. Springer.

[7] Sanjoy Dasgupta, Christos Papadimitriou, and Umesh Vazirani, *Algorithms*, 2008, Chapters 8, 9.

[8] Stephen Cook, *The complexity of theorem-proving procedures, Proceedings of the 3rd Annual ACM Symposium on Theory of Computing,* pages 151-158, 1971.

[9] Goldreich, Oded, *P, NP, and NP-Completeness: The basics of Computational Complexity*, Cambridge University press, 2010, Chapters 1, 2, 4.

[10] Frederick S Hillier, Gerald J Lieberman, *Introduction to Operations Research*, 5th Edition, McGraw-Hill, Chapter 13, 1990.

[11] Glover Fred, *Tabu Search – Part I*, ORSA Journal on Computing, Volume 1, Issue 3, pp. 190-206, 1989.

[12] Glover Fred, Tabu Search – Part II, ORSA Journal on Computing, Volume 2, Issue 1, pp. 4-32, 1990.

[13] Manuel Laguna, *A guide to implementing Tabu Search*, Volume 4, No 1, Investigación Operativa, 1994.

[14] Fred Glover, *Tabu Search: A Tutorial,* Interfaces, 1990.

[15] Qingqiang Guo, Katagiri H., *A hybrid algorithm based on tabu search and immune algorithm for k - cardinality tree problems,* Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS), 2012 Joint International Conference, pp. 346 – 351.

[16] Fred Glover, *Tabu thresholding: Improved search by nonmonotonic trajectories,* ORSA journal on computing, Volume 7, No. 4, pp. 426 – 442, 1995.

[17] John Knox, *Tabu Search Performance on the Symmetric Travelling Salesman Problem,* Computers and Operations Research, Volume 21, Issue 8, pp. 867 – 876. 1994.

[18] Sumantha Basu, *Tabu Search Implementation on Travelling Salesman Problem and Its Variations: A Literature Survey,* American Journal of Operations Research, Volume 2, Issue 2, pp. 163 – 173, 2012.

[19] Knox, John Edward, *The Application of Tabu Search to the Symmetric Travelling Salesman Problem*, PhD dissertation, Graduate School of Business, University of Colorado, July 1989.

[20] Sumanta Basu, Ravindra S. Gajulapalli, Diptesh Ghosh, *A Fast Tabu Search Implementation for large Asymmetric Travelling Salesman Problems defined on Sparse Graphs*, OPSEARCH, Volume 50, Issue 1, pp. 75 – 88, 2013.

[21] Fred Glover, *Artificial Intelligence, Heuristic Frameworks and Tabu Search*, Managerial and Decision Economics, Volume 11, pp. 365 – 375, 1990.

[22] Knox, J. and Glover, F., *Comparative testing of Travelling Salesman Heuristics derived from Tabu Search, Genetic Algorithms and Simulated Annealing*, Center of Applied Artificial Intelligence, University of Colorado, September 1989.

[23] Ning Yang, Ping Li, Baisha Mei, *An angle-based crossover tabu search for travelling salesman problem,* Natural Computation, 2007, ICNC 2007, Third International Conference (Volume: 4), pp. 512 – 516, Conference dates 24 – 27 August 2007.

[24] Yi He, Yuhui Oiu, Guangyuan Liu, Kaiyou Lei, *A parallel adaptive tabu search approach for travelling salesman problem,* Natural Language Processing and Knowledge Engineering, 2005, IEEE NLP-KE 2005, IEEE International Conference, pp. 796 – 801.

[25] Niizuma Daichi, Yasuda Keiichiro, Ishigame Atsushi, *Multi-point tabu search for travelling salesman problem,* IEEJ Transactions on Electrical and Electronic Engineering, Volume 1, Issue 1, pp. 126 – 129, 2006.

[26] Yiwen Zhong, Chao Wu, Lishan Li, Zhengyuan Ning, *The study of Neighborhood structure of tabu search algorithm for travelling salesman problem,* Natural Computation, 2008, ICNC 2008, Fourth International Conference (Volume :1), pp. 491 – 495.

[27] Fred Glover, *Tabu search and adaptive memory programming – Advances, applications and challenges.*

[28] Brucker, Peter. *Scheduling Algorithms*, Chapter 1, 6. Springer, 2007.

[29] R.L. Graham, E.L.Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, *Optimization and approximation in deterministic sequencing and scheduling: a survey*, Ann. Discr. Math, pp. 287 – 326, 1979.

[30] Mauro Dell'Amico and Marco Trubian, *Applying Tabu Search to the Job-Shop Scheduling Problem*, Annals of Operations Research, Volume 41, Issue 3, pp. 231 – 252, 1993.

[31] Eugeniusz Nowicki and Czeslaw Smutnicki, *A Fast Taboo Search Algorithm for Job Shop Problem*, Volume 42, Issue 6, pp. 797 – 813, 1996.

[32] S. G. Ponnambalam, P. Aravindan and S. V. Rajesh, *A Tabu Search Algorithm For Job Shop Scheduling*, International Journal of Advanced Manufacturing Technology, Volume 16, Issue 10, pp. 765 – 771, 2000.

[33] J. Wesley Barnes, John B. Chambers, *Solving the job shop problem with tabu search*, IIE Transactions, Volume 27, Issue 2, pp. 257 – 263, 1995.

[34] Song, Ju-seog and Lee, Tae-Eog, *A Tabu Search Procedure for Periodic Job Shop Scheduling*, Computers and Industrial Engineering, Volume 30, Issue 3, pp. 433 – 447, 1996.

[35] Manuel Laguna, Rafael Marti, Vicente Campos, *Intensification and Diversification with elite tabu search solutions for the linear ordering problem*, March 1998.

[36] Saidi-Mehrabad, Mohammad and Fattahi, Parviz, *Flexible Job Shop Scheduling with Tabu Search Algorithms*, The International Journal of Advanced Manufacturing Technology, Volume 32, Issue 5, pp. 563 – 570, 2007.

[37] Meeran, S and Morshed, M S, *A Hybrid Genetic Tabu Search Algorithm for solving Job Shop Scheduling Problems*, Journal of Intelligent Manufacturing, Volume 23, Issue 4, pp. 1063 – 1078, 2012.

[38] Gouhui Zhang, Yang Shi, Liang Gao, *A genetic algorithm and tabu search for solving flexible job shop schedules*, computational Intelligence and Design, 2008, ISCID 2008, International Symposium on (Volume :1), pp. 369 – 372.

[39] Yajie Tian, Sannomiya N., Yuedong Xu, *A tabu search with a new neighborhood search technique applied to flow shop scheduling problems,* Decision and Control, 2000, 39th IEEE Conference (Volume :5), pp. 4606 – 4611.

[40] Tamura S, Kawamura N, Ikkai Y, Komoda N, *An interactive high speed scheduling method by tabu search for a large scale job shop problem with group constraints*, Emerging Technologies and Factory Automation, 2005, 10th IEEE Conference, pp. 826.

VITA

Graduate College

University of Nevada, Las Vegas

Lemasri Piniganti

Degrees:

Bachelor of Technology in Information Technology, 2011

Jawaharlal Nehru Technological University

Master of Science in Computer Science, 2013

University of Nevada Las Vegas

Thesis Title: A Survey of Tabu Search in Combinatorial Optimization

Thesis Examination Committee:

Chair Person, Dr. Wolfgang Bein, Ph.D.

Committee Member, Dr. Ajoy K. Datta, Ph.D.

Committee Member, Dr. Ju-Yeon Jo, Ph.D

Graduate College Representative, Dr. Venkatesan Muthukumar, Ph.D.