Tabu search algorithm

UNIT IV

Tabu Search: Features

- Tabu search is a Meta heuristic (Metaheuristics are strategies that guide the search process. · The goal is to efficiently explore the search space in order to find near–optimal solutions.) loosely connected to evolutionary computing. It is designed to overcome difficult regions of a search space by imposing restrictions.
- Tabu search uses memory – short term, long term and intermediate – to achieve diversification and intensification.
- It can be used to solve combinatorial optimization problems where a solution to a discrete problem with the presence of constraints or decision variables.
- The principal characteristic of tabu search is based on using a mechanism which is inspired by the human memory
  - 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.
- 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.

Tabu Search: Problem definition

- Originally Tabu search was proposed by Fred Glover in 1986 to overcome the local optima that are faced by the local search algorithms. It is a kind of iterative search and is characterized by the use of a flexible memory.
- It is able to eliminate local minima and to search areas beyond a local minimum. Therefore, it has the ability to find the global minimum of a multimodal search space.

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 ϵ 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.

Tabu Search: Concept

- 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 suited to the given problem

Tabu Search: Concept

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.
  - Explicit 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.
  - Attributive 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.
- This memory reduces the size of the neighbourhood by forbidding some moves in the search. In tabu search algorithm we use both explicit and attributive memory

Tabu Search: Concept

Types of memory structures

There are two types of memory structures: short term memory and long term memory.

The memory structures are functioned by referring to four principal dimension based memory:

- Recency
  - 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
  - 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.
- Quality
  - The Quality based memory is used for the ability to differentiate the merit of solutions visited during the search.
- Influence.
  - The Influence based memory is used to impact the choices made during the search. It focuses on quality as well as structure.

Tabu Search: Concept

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 neighbourhood 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 neighbourhood path of the elite solutions by integrating the intensification and diversification strategies.

Tabu Search: Algorithm

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.

Tabu Search: Algorithm

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 neighbourhood 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.

Tabu Search: Algorithm

- **Creating a candidate list**
- After obtaining the initial solution find all the possible moves from the neighbourhood of the current solution that can be applied on the initial solution.
- Before selecting the moves define the neighbourhood 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.

Tabu Search: Algorithm

- **Creating a candidate list**
- 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.

Tabu Search: Algorithm

```
algorithm tabu search
begin
  tabu_list:= [];
  S:= initial solution;
  S*:= S;
  Repeat
    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;
```

Tabu Search: Algorithm

- **Creating a candidate list**
- **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 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.

Tabu Search: Algorithm

- **Creating a candidate list**
- **Tabu list and Aspiration levels**
- 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.

Tabu Search: Algorithm

- **Creating a candidate list**
- **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.

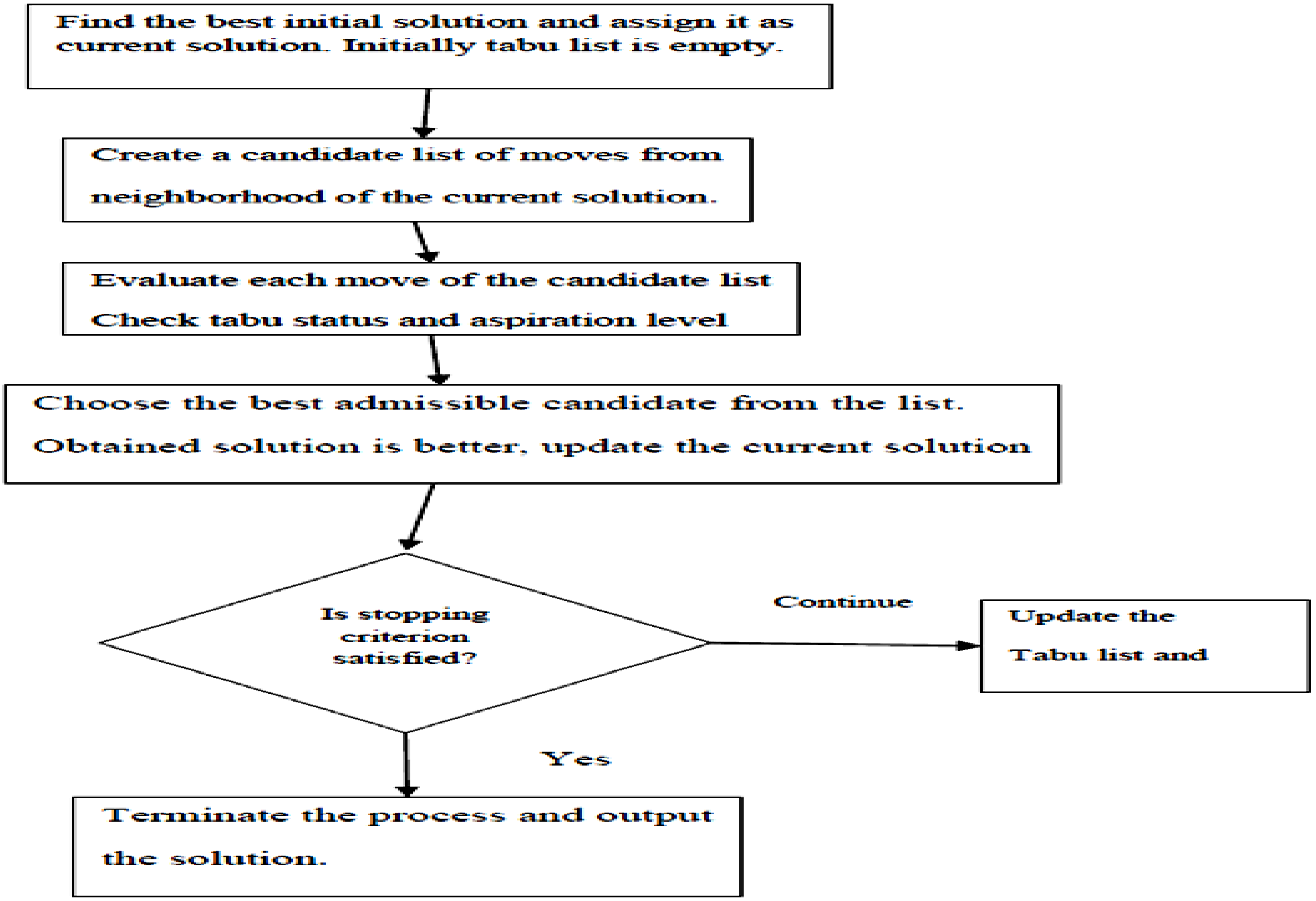
Tabu Search: Algorithm

- **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.

Tabu Search: Flow Chart

- 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.

Tabu Search: Flow Chart


Find the best initial solution and assign it as current solution. Initially tabu list is empty.
Create a candidate list of moves from neighborhood of the current solution.
Evaluate each move of the candidate list
Check tabu status and aspiration level
Choose the best admissible candidate from the list.
Obtained solution is better, update the current solution
Is stopping criterion satisfied?
Continue
Update the Tabu list and
Yes
Terminate the process and output the solution.

Tabu Search:

Tabu Search is a commonly used meta-heuristic used for optimizing model parameters.

A meta-heuristic is a general strategy that is used to guide and control actual heuristics.

Tabu Search is often regarded as integrating memory structures into local search strategies.

As local search has a lot of limitations, Tabu Search is designed to combat a lot of those issues.

The basic idea of Tabu Search is to penalize moves that take the solution into previously visited search spaces (also known as tabu).

Tabu Search, however, does deterministically accept non-improving solutions in order to prevent getting stuck in local minimums.

Tabu Search:

Short-term vs. Long-term Memory

Short Term memory

- It is based off of recency of occurrence and is used to prevent the search algorithm from revisiting previously visited solutions and also can be used to return to good components in order to localize and intensify a search. This is accomplished by the Tabu List and is also known as intensification.

Long Term memory

- It is based off of frequency of occurrence and is used to diversity the search and explore unvisited areas of the search space by avoiding explored areas. This is accomplished by frequency memory and is also known as diversification.

Tabu Search:

Tabu List

The Tabu List is the cornerstone of utilizing short-term memory. This list stores a fixed amount of recently made moves.

In some implementations, complete solutions are used instead of the moves used but this is not ideal if complete solutions are very large due to space limitations.

Some examples of these moves are:

1. swap nodes in a graph/tour
2. toggling a bit between 0 and 1
3. insert or delete edges in a graph

Tabu Search:

Tabu Tenure

The Tabu Tenure is the number of iterations that a move stays in the Tabu List. The moves that are in the Tabu List are moves that cannot be made again as it was already recently visited. There are two ways to implement the Tabu Tenure (T):

Static: choose T to be a constant (often sqrt(T))

Dynamic: choose T randomly between some T_min and T_max

Tabu Search:

Aspiration Criteria

This is an optional part of Tabu Search. Moves or solutions that are part of the Aspiration Criteria cancel out the Tabu and the move can be made even if it’s in the Tabu List. This can also be used to prevent stagnation in cases where all possible moves are prohibited by the Tabu List. Some examples of Aspiration Criteria are:

- if the new solution is better than the current best solution, then the new solution is used even if it’s on the Tabu List
- setting the Tabu Tenure to be a smaller value

Tabu Search:

Frequency Memory

This memory holds the total number of iterations that each solution was picked since the beginning of the search. Solutions that were visited more are less likely to to be picked again and would promote more diverse solutions. There are two main approaches for diversifying:

Restart Diversification: allows components that rarely appear to be in the current solution by restarting the search from these points

Continuous Diversification: biases the evaluation of possible moves with the frequency of these moves. Moves that do not appear as often will then have a higher probability to be made.

Tabu Search:

- Algorithm
- **Step 1:** We first start with an initial solution ***s = S₀***. This can be any solution that fits the criteria for an acceptable solution.
- **Step 2:** Generate a set of neighbouring solutions to the current solution ***s*** labeled ***N(s)***. From this set of solutions, the solutions that are in the Tabu List are removed with the exception of the solutions that fit the Aspiration Criteria. This new set of results is the new ***N(s)***.

$$s' \in N(s) = \{N(s) - T(s)\} + A(s)$$

Tabu Search:

- Algorithm
- **Step 3:** Choose the best solution out of ***N(s)*** and label this new solution ***s'***. If the solution ***s'*** is better than the current best solution, update the current best solution. After, regardless if ***s'*** is better than ***s***, we update ***s*** to be ***s'***.
- **Step 4:** Update the Tabu List ***T(s)*** by removing all moves that are expired past the Tabu Tenure and add the new move ***s'*** to the Tabu List. Additionally, update the set of solutions that fit the Aspiration Criteria ***A(s)***. If frequency memory is used, then also increment the frequency memory counter with the new solution.

Tabu Search:

- Algorithm
- **Step 5:** If the Termination Criteria are met, then the search stops or else it will move onto the next iteration. Termination Criteria is dependent upon the problem at hand but some possible examples are:
- a max number of iterations
- if the best solution found is better than some threshold

Tabu Search:

Examples of Problems to Solve with TS

1. N-Queens Problem
2. Traveling Salesman Problem (TSP)
3. Minimum Spanning Tree (MST)
4. Assignment Problems
5. Vehicle Routing
6. DNA Sequencing

Tabu Search:

Advantages:

1. Can escape local optimums by picking non-improving solutions
2. The Tabu List can be used to avoid cycles and reverting to old solutions
3. Can be applied to both discrete and continuous solutions

Disadvantages:

1. Number of iterations can be very high
2. There are a lot of tuneable parameters in this algorithm

Tabu Search:

Conclusion:

- Tabu Search is a popular algorithm used to optimize a multi-parameter model that can yield exceptional results.
- Although the implementation is not trivial and requires tuning, it is capable of solving a wide variety of problems once it is created.