Tabu Search Scheduling

Heansuh Lee
Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
heansuh.lee@stud.hshl.de

***Abstract*—Tabu Search Scheduling is a heuristic optimization technique used to solve scheduling problems by exploring the solution space and selecting the best solutions according to a specific objective function. It is characterized by the use of a ”tabu list,” which enables the algorithm to escape local optima and explore a wider solution space. Tabu Search Scheduling has several variants and has been applied to various scheduling problems, showing effectiveness in improving schedule quality and resource utilization. This paper gives an overview of Tabu Searching Scheduling algorithm, and discusses its current applications, the strengths and weaknesses, and its evaluation in future applications. There will be an overview that tabu search scheduling algorithm is used in the High Level Synthesis (HLS).**

I. Tabu Search Scheduling

A. Introduction

Tabu Search Scheduling is a heuristic optimization technique used to solve scheduling problems in various fields such as production planning, transportation, and project management. It is based on the concept of exploring the solution space by iteratively generating candidate solutions and selecting the best ones according to a specific objective function [1].

Tabu Search Scheduling is characterized by the use of a ”tabu list,” which is a memory structure that stores information about previously visited solutions and prevents revisiting them during the search process. This mechanism enables the algorithm to escape local optima and explore a wider solution space in search of a global optimum.

Tabu Search Scheduling has been applied to various scheduling problems, such as job shop scheduling, flow shop scheduling, and vehicle routing. It has been shown to be effective in improving the quality of schedules, reducing production time, and enhancing resource utilization.

In conclusion, Tabu Search Scheduling is a heuristic optimization technique used to solve scheduling problems by exploring the solution space and selecting the best solutions according to a specific objective function. It is characterized by the use of a tabu list, which enables the algorithm to escape local optima and explore a wider solution space. Despite its limitations, Tabu Search Scheduling has been shown to be effective in improving the quality of schedules and resource utilization in various domains [2].

B. Definition

In the context of the Tabu Search algorithm, ”Tabu” refers to a restriction, or a rule that certain solutions or moves are ”forbidden”. The term ”Tabu” is derived from the Tongan word ”Taboo”, which implies something that is forbidden or not allowed.

The primary idea behind the Tabu Search algorithm is to avoid getting stuck in local optimal solutions while searching for the global optimal solution. To achieve this, the algorithm keeps a record of the last few solutions or moves in the search space, known as the ”Tabu List”. When considering the next move, the algorithm will not consider any solution that appears in the Tabu List, effectively prohibiting the search from revisiting recent solutions.

The goal of this strategy is to encourage exploration of the search space, helping the algorithm to escape local optima and increasing the chances of finding the global optimum. To balance exploration and exploitation, the algorithm will also include an aspiration criterion, which can override the tabu status of a move if it results in a solution better than any previously encountered.

In the context of scheduling in High-Level Synthesis (HLS), each ”solution” would be a particular schedule of operations, and each ”move” would involve reordering some of the operations in the schedule. The Tabu List would keep track of recent schedules or moves to avoid repetitively exploring the same or similar schedules [3].

C. Main Strategies

Forbidding, freeing, and short-term strategies — are essential components of the Tabu search algorithm that determine how the search explores and exploits the solution space [4].

- **Forbidding strategy:** The forbidding strategy is implemented through the use of a tabu list, which records recently visited solutions or applied local moves. The main purpose of the tabu list is to prevent the search from cycling back to previously visited solutions, thus encouraging the exploration of new regions in the search space. By marking certain moves or solutions as ”tabu” or forbidden for a specific number of iterations, the algorithm avoids getting stuck in local optima and promotes diversification.
- **Freeing strategy:** The freeing strategy, also known as aspiration criteria, provides a mechanism to override the tabu status of certain candidate solutions under specific conditions. This strategy helps balance the exploration and exploitation trade-off in the search process. By allowing the algorithm to consider tabu solutions that meet certain aspiration criteria (e.g., better objective value

or problem-specific conditions), the search process can avoid becoming overly focused on a narrow region of the search space and explore more diverse solutions. This ultimately helps the Tabu search algorithm to find high-quality solutions to complex combinatorial optimization problems.

- **Short-term strategy:** The short-term strategy in Tabu search refers to the use of short-term memory to guide the search process. The tabu list represents a form of short-term memory, as it stores recently visited solutions or local moves for a limited number of iterations. This short-term memory helps the algorithm to avoid cycling and encourages the exploration.

D. Aspiration Criteria

Aspiration criteria play a crucial role in the Tabu search algorithm by allowing the search process to override the tabu status of certain candidate solutions under specific conditions. This mechanism helps the algorithm avoid becoming trapped in local optima and promotes exploration of promising regions in the search space [5].

In the context of the Tabu search algorithm, aspiration criteria are conditions that permit the selection of a candidate solution even if it is marked as tabu. This is usually done when the candidate solution exhibits particularly promising characteristics, such as:

- **Better objective value:** If the candidate solution has a better objective value (e.g., shorter tour length in TSP or lower cost in hardware-software co-design scheduling) than the current best-known solution, the algorithm may override the tabu status and consider it for the next iteration. This encourages the algorithm to pursue more promising regions in the search space.
- **Sufficient time since the last visit:** If a certain amount of time (measured in iterations) has passed since a solution was last visited, it may be worthwhile to reconsider it. This allows the algorithm to revisit earlier promising solutions, which might have been marked as tabu at the time, but could now provide new insights or lead to better solutions.
- **Other problem-specific criteria:** Depending on the problem being solved, additional aspiration criteria can be designed based on the specific characteristics of the problem. For example, in scheduling problems, a candidate solution might be considered if it significantly reduces the completion time of a critical task, even if it is marked as tabu.

Aspiration criteria play a key role in balancing the exploration and exploitation trade-off in the Tabu search algorithm. By occasionally allowing the algorithm to consider tabu solutions that meet certain aspiration criteria, the search process can avoid becoming overly focused on a narrow region of the search space and explore more diverse solutions. This ultimately helps the Tabu search algorithm to find high-quality solutions to complex combinatorial optimization problems like TSP and hardware-software co-design scheduling [6] [7].

E. Neighbourhood Search Methods

Neighborhood search methods involve exploring a set of candidate solutions in the vicinity of the current solution. The goal is to find a better solution by applying small changes or local moves to the current solution.

The neighborhood structure is a crucial aspect of neighborhood search methods. It defines the set of candidate solutions that can be reached from the current solution by applying a local move. The choice of neighborhood structure has a significant impact on the efficiency and effectiveness of the search process. Different neighborhood structures can be employed depending on the problem being solved.

Some examples of neighborhood structures for common optimization problems:

Traveling Salesman Problem (TSP):

- **2-opt:** This involves swapping the endpoints of two edges in the current tour, resulting in a new tour.
- **3-opt:** This involves removing three edges from the current tour and reconnecting the remaining fragments in a different order to create a new tour.
- **Or-opt:** This involves relocating a sequence of consecutive cities within the tour to a different position.

Hardware-software co-design scheduling:

- **Task swapping:** This involves swapping the assignment of two tasks between hardware and software components.
- **Task reassignment:** This involves reassigning a single task from a hardware component to a software component or vice versa.
- **Task group reassignment:** This involves reassigning a group of related tasks from hardware components to software components or vice versa.

In Tabu search, the neighborhood search process involves generating the set of candidate solutions in the neighborhood of the current solution, evaluating their objective values (e.g., tour length in TSP or cost in hardware-software co-design), and selecting the best non-tabu candidate as the next solution. The tabu list is then updated to record the recent local moves and prevent cycling.

The neighborhood search method is essential to the success of Tabu search, as it guides the exploration and exploitation of the search space. By defining an appropriate neighborhood structure and searching it efficiently, Tabu search can effectively find near-optimal solutions for a wide range of combinatorial optimization problems.

II. Relation to High Level Synthesis

A. Travelling Salesman Problem

Tabu search scheduling algorithm can be applied to various combinatorial optimization problems, including scheduling in the context of hardware-software co-design and the Traveling Salesman Problem (TSP) [8].

In hardware-software co-design, the objective is to find an optimal partitioning of a system's functionality between hardware and software components. The main goal is to minimize the overall cost, which can include metrics such as design time, power consumption, and performance. Tabu search can be employed to explore the design space efficiently and find near-optimal solutions for partitioning tasks between hardware and software components.

The Traveling Salesman Problem (TSP) is a classic combinatorial optimization problem that asks the following question: Given a list of cities and the distances between them, what is the shortest possible route that visits each city exactly once and returns to the origin city? TSP is an NP-hard problem, meaning that it is computationally difficult to find an optimal solution in a reasonable amount of time for large instances.

Tabu search can be used to solve the TSP by iteratively searching for better solutions in the neighborhood of the current solution while avoiding revisiting recently explored solutions. The main components of the Tabu search algorithm are:

- **Solution representation:** In the TSP context, a solution can be represented as a permutation of the cities, indicating the order in which they are visited.
- **Neighborhood structure:** This defines the set of candidate solutions that can be reached from the current solution by applying a local move (e.g., swapping the position of two cities in the permutation).
- **Tabu list:** This is a short-term memory that stores recently visited solutions or local moves to prevent cycling and encourage exploration of new regions in the search space.
- **Aspiration criteria:** These are conditions that allow the algorithm to override the tabu status of a solution if it is particularly promising (e.g., if it is better than the current best-known solution).
- **Stopping criteria:** The algorithm stops when a specified number of iterations have been performed or another condition is met (e.g., no improvement in the best solution for a certain number of iterations).

Tabu search explores the solution space by repeatedly applying local moves to the current solution, selecting the best non-tabu candidate from the neighborhood, and updating the tabu list. The algorithm typically employs diversification strategies to escape local optima and intensification strategies to exploit promising regions in the search space. By balancing exploration and exploitation, Tabu search can effectively find near-optimal solutions to complex optimization problems like TSP and hardware-software co-design scheduling.

III. Applications

Here are some applications of Tabu Search scheduling algorithm.

A. Vehicle Routing Problem (VRP)

The Vehicle Routing Problem (VRP) is a classic problem in the fields of transportation, distribution, and logistics. The problem involves designing optimal routes for a fleet of vehicles to deliver goods to a set of customers. The objective is typically to minimize total travel distance, time, or cost, subject to various constraints [9].

A neighborhood search strategy is employed here, where a ”neighborhood” is defined by a set of solutions that can be reached by swapping two customers between routes or moving a customer from one route to another.

The algorithm starts with an initial solution and iteratively moves to a neighboring solution that improves the objective function. To avoid getting trapped in local optima, the algorithm uses a tabu list, which is a short-term memory structure that keeps track of recent moves and forbids them for a certain number of iterations.

It tests the algorithm on a set of benchmark problems and found that it performed very well, often finding the best known solutions and sometimes finding new best solutions. The conclusion is that that Tabu Search is a very effective method for the VRP.

It’s important to note that the VRP is a complex and diverse problem, with many variations depending on the specific constraints and objectives. For example, there are versions of the VRP with time windows (where each customer has a specific time window during which they must be served), the Capacitated VRP (where each vehicle has a maximum capacity), and many others. Different versions of the problem may require different solution methods or adaptations of the basic Tabu Search algorithm.

B. Job Shop Scheduling

The Job Shop Scheduling problem is a classic problem in operations research and production management. The problem involves scheduling a set of jobs on a set of machines, with the goal of optimizing a certain objective, such as minimizing total completion time, lateness, or makespan (the total time from start to finish) [10].

The algorithm uses a neighborhood search strategy, where a ”neighborhood” is defined by a set of solutions that can be reached by swapping two operations in a schedule. The algorithm starts with an initial solution and iteratively moves to a neighboring solution that improves the objective function. To avoid getting trapped in local optima, the algorithm uses a tabu list, which is a short-term memory structure that keeps track of recent moves and forbids them for a certain number of iterations.

Aspiration criteria that allows the algorithm is introduced, to override the tabu status of a move if it leads to a solution better than any previously found. This helps to further diversify the search and avoid local optima.

The Tabu Search algorithm was tested on a set of benchmark problems and found that it performed very well, often finding the best known solutions and sometimes finding new best solutions. So, Tabu Search scheduling algorithm is a very effective method for the Job Shop Scheduling problem.

The Job Shop Scheduling problem is a complex problem with many variations depending on the specific constraints and objectives. For example, there are versions of the problem with sequence-dependent setup times, machine availability constraints, and many others. Different versions of the problem may require different solution methods or adaptations of the basic Tabu Search algorithm.

C. Network Design

Network design is a critical aspect of telecommunications, transportation, and logistics, where the goal is to design a network that optimizes a certain objective, such as minimizing total cost or maximizing service level, subject to various constraints [11].

In the context of the paper ”A simplex-based tabu search method for capacitated network design”, the problem involves designing a capacitated network, where each link in the network has a certain capacity that cannot be exceeded. The objective is typically to minimize the total cost of installing and operating the network, subject to the capacity constraints and the requirement to meet a certain level of demand.

The Tabu Search algorithm is used to solve this problem by iteratively exploring the solution space, which consists of different network designs. Each solution, or network design, is represented by a set of decision variables indicating whether each possible link is installed and how much flow is sent over each link.

A ”neighborhood” of a solution is defined by a set of solutions that can be reached by making small changes to the current solution, such as adding or removing a link, or changing the flow on a link. The algorithm starts with an initial solution and iteratively moves to a neighboring solution that improves the objective function.

To avoid getting trapped in local optima, the algorithm uses a tabu list, which is a short-term memory structure that keeps track of recent changes and forbids them for a certain number of iterations. This encourages the algorithm to explore different parts of the solution space and increases the chances of finding a global optimum.

Aspiration criteria allows the algorithm to override the tabu status of a change if it leads to a solution better than any previously found. This helps to further diversify the search and avoid local optima.

The simplex method, a popular algorithm for solving linear programming problems, is used to solve the subproblem of determining the flow on each link given a network design. This makes the algorithm more efficient and allows it to handle larger problems.

The algorithm has been found to be very effective for the capacitated network design problem, often finding high-quality solutions in a reasonable amount of time. It is a flexible method that can be adapted to different variations of the problem and different types of network design problems.

D. Feature Selection in Machine Learning

Feature selection is a critical step in the machine learning pipeline. It involves selecting a subset of the most informative features (or variables) to use in model training. The goal is to improve the model's performance by reducing overfitting, improving accuracy, and reducing training time [12].

In the context of the paper ”Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power”, the Tabu Search algorithm is used to solve the feature selection problem.

Each solution in the search space is a subset of features. A ”neighborhood” of a solution is defined by the sets of features that can be obtained by adding or removing a feature from the current set. The algorithm starts with an initial solution and iteratively moves to a neighboring solution that improves the objective function, which is typically a measure of the predictive performance of a machine learning model trained on the selected features.

To avoid getting trapped in local optima, the algorithm uses a tabu list, which is a short-term memory structure that keeps track of recent changes (i.e., features that have been added or removed) and forbids them for a certain number of iterations. This encourages the algorithm to explore different parts of the solution space and increases the chances of finding a global optimum.

Aspiration criteria allows the algorithm to override the tabu status of a change if it leads to a solution better than any previously found. This helps to further diversify the search and avoid local optima.

In the context of feature selection, the Tabu Search algorithm has been found to be very effective, often finding subsets of features that lead to high-performing machine learning models. It is a flexible method that can be adapted to different types of machine learning problems and different measures of predictive performance.

IV. Implementation in Python

This is a very basic version of the Tabu Search algorithm. Here's a brief explanation of the components:

- **initial_solution**: This is the starting point for the search. It could be any feasible solution to the problem.
- **objective_function**: This is a function that takes a solution and returns a score indicating how good the solution is. The goal of the Tabu Search is to find the solution that minimizes this score.
- **neighborhood_function**: This is a function that takes a solution and returns a list of its ”neighbors”. In the context of a scheduling problem, a neighbor might be a schedule that is similar to the current one but with one or two jobs swapped.
- **max_iterations**: This is the maximum number of iterations that the algorithm will run for. After this many iterations, the algorithm will stop and return the best solution it has found.
- **tabu_list_size**: This is the maximum size of the tabu list. The tabu list is a list of solutions that the algorithm has visited recently and is not allowed to revisit. This helps the algorithm avoid getting stuck in cycles.

```
Input  : initial_solution, objective_function,
         neighborhood_function, max_iterations,
         tabu_list_size
Output: best_solution
current_solution ← initial_solution;
best_solution ← current_solution;
tabu_list ← [];
for i in range(max_iterations) do
    neighbors ←
     neighborhood_function(current_solution);
    neighbors ← [neighbor for neighbor in neighbors
     if neighbor not in tabu_list];
    if not neighbors then
        neighbors ←
         neighborhood_function(current_solution);
    end
    best_neighbor ← min(neighbors,
     key=objective_function);
    current_solution ← best_neighbor;
    if objective_function(best_neighbor) ¡
     objective_function(best_solution) then
        best_solution ← best_neighbor;
    end
    tabu_list.append(best_neighbor);
    if len(tabu_list) ¿ tabu_list_size then
        tabu_list.pop(0);
    end
end
```

Algorithm 1: Tabu Search Algorithm

V. Evaluation

Tabu Search (TS) Scheduling algorithm is a metaheuristic algorithm that has been effectively applied to solve various complex optimization problems, including scheduling. It is renowned for its flexibility and robustness, often providing high-quality solutions. The algorithm's memory-based mechanism is a distinctive feature that helps it avoid entrapment in local optima by prohibiting or penalizing moves that lead to recently visited solutions. This characteristic enables TS to explore the solution space more thoroughly and escape local optima, which is a common pitfall of many optimization algorithms. Furthermore, TS can handle a wide variety of problem types, including those with complex constraints and nonlinear objective functions [13] [14].

However, TS is not without its limitations. The performance of the algorithm can be highly dependent on the proper tuning of its parameters, such as the tabu list size and the neighborhood structure. This tuning process can be time-consuming and requires a good understanding of the problem at hand. Additionally, TS may require a significant amount of computational resources, especially for large-scale problems. While TS often finds high-quality solutions, it does not guarantee finding the optimal solution, especially for very complex problems.

To use TS effectively, several strategies can be employed. The choice of the initial solution can have a significant impact on the performance of the algorithm. A good initial solution can guide the search towards promising regions of the solution space. The neighborhood structure, which determines which solutions are considered for the next move at each step of the algorithm, should be defined in a way that allows a good balance between exploration (searching new areas of the solution space) and exploitation (refining the current best solution). The size of the tabu list, which stores recently visited solutions, should be chosen carefully. A too small tabu list may not prevent the algorithm from revisiting solutions, while a too large one may limit the search space excessively.

Comparatively, Simulated Annealing (SA) is another popular metaheuristic algorithm. Like TS, SA is also capable of escaping local optima, but it does so using a probabilistic mechanism based on the metaphor of annealing in metallurgy. SA accepts worse solutions with a certain probability, which decreases over time. This allows SA to explore the solution space widely in the early stages of the search and then gradually focus on exploitation as the algorithm progresses.

While both TS and SA have proven effective for solving complex optimization problems, they have different strengths and weaknesses. SA's main advantage is its simplicity and ease of implementation. However, its performance can be highly sensitive to the cooling schedule, which determines how the probability of accepting worse solutions decreases over time. On the other hand, TS's use of a memory structure can provide a more guided and effective search, but at the cost of increased complexity and computational requirements.

Both TS and SA are powerful tools for solving optimization problems. The choice between them depends on the specific characteristics of the problem and the resources available.

VI. Conclusion

In conclusion, the Tabu Search scheduling algorithm is a powerful metaheuristic approach for solving complex optimization problems, including scheduling. Its strength lies in its ability to escape local optima through a memory-based mechanism, which allows it to explore the solution space more thoroughly. However, its performance can be sensitive to parameter tuning and it may require significant computational resources. Compared to other metaheuristics like Simulated Annealing, TS offers a more guided search strategy, albeit at the cost of increased complexity. Despite its limitations, TS has proven to be a flexible and robust tool for a wide variety of problem types, making it a valuable addition to the toolbox of optimization techniques.

References

[1] J. Barnes, M. Laguna, and F. Glover, *An Overview of Tabu Search Approaches to Production Scheduling Problems*, ser. Operations Research / Computer Science Interfaces Series. Springer, 1995, vol. 3.

[2] F. Glover, J. P. Kelly, and M. Laguna, "Genetic algorithms and tabu search: Hybrids for optimization," *Computers & Operations Research*, vol. 22, no. 1, pp. 111–134, 1995.

[3] S. Hanafi, "On the convergence of tabu search," *Journal of Heuristics*, vol. 7, pp. 47–58, 2001.

[4] F. Glover, “Tabu search: A tutorial,” *Interfaces*, vol. 20, pp. 74–94, 08 1990.

[5] F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*, 8th ed. New York, NY: McGraw-Hill, 2005.

[6] M. Ji and H. Tang, “Global optimizations and tabu search based on memory,” *Applied Mathematics and Computation*, vol. 159, pp. 449–457, 2004.

[7] C. R. Reeves, *Modern Heuristic Techniques for Combinatorial Problems*. John Wiley & Sons, Inc., 1993.

[8] D. T. Pham and D. Karaboga, *Intelligent Optimisation Techniques–Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*. London: Springer-Verlag, 2000.

[9] M. Gendreau, A. Hertz, and G. Laporte, “A tabu search heuristic for the vehicle routing problem,” *Management science*, vol. 40, no. 10, pp. 1276–1290, 1994.

[10] E. Nowicki and C. Smutnicki, “A fast taboo search algorithm for the job shop problem,” *Management science*, vol. 42, no. 6, pp. 797–813, 1996.

[11] T. G. Crainic, B. Gendron, and J. M. Farvolden, “A simplex-based tabu search method for capacitated network design,” *INFORMS Journal on Computing*, vol. 10, no. 3, pp. 341–353, 1998.

[12] S. García, A. Fernández, J. Luengo, and F. Herrera, “Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power,” *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, 2010.

[13] “Comprehensive survey on tabu search algorithms (tsa),” *IEEE Xplore*, 2020, accessed: 05/06/2023. [Online]. Available: https://ieeexplore.ieee.org/document/9091743

[14] F. Glover and M. Laguna, *Tabu Search*. Norwell, MA: Kluwer Academic Publishers, 1997.