Cadi Ayyad University

2023/2024

Tabu Search

Presented by:
ARJADAL Moncef
ASRI Bachir
IGMOULLAN Iliass

Supervised by:
Pr. **OUAARAB** Aziz

Table of Contents

Introduction

Local search ends up in a local optimum: we need to escape from local optima in order to search the solution space more extensively and effectively.

• Any iterative search/exploration process should accept also non-improving moves to be able to escape from a local optimum.

• As soon as non-improving moves are possible, there is the risk of visiting again a solution, falling back to a previous local optimum, or, more generally, cycling ! waste of resources

Introduction

Tabu search (TS) (Fred Glover, 1989) is an iterative, memory-based neighborhood-search method.

• The optimization process proceeds by generating a sequence of solutions by applying some local neighborhood search.

• TS keeps information on the itinerary through the visited solutions.

• Such information is used to guide the move from a solution point ***s*** to the next by restricting the possible local choices to some subset of ***N(s)*** or by modifying the neighborhood itself (by locally modifying the objective function)

What is Tabu Search?

Tabu Search is a popular and versatile optimization technique used to solve complex problems. It falls under the category of local search algorithms and is particularly well-suited for combinatorial optimization problems.

The term "tabu" originates from a Polynesian word meaning "forbidden" or "prohibited," and this concept plays a central role in Tabu Search.

What is Tabu Search?

- Tabu Search uses a local search approach.
- Iteratively moves from one solution (x) to an improved solution (x') within its neighborhood.
- TS Continues this process until certain conditions are met (e.g., predefined attempt limit or score threshold).
- TS Carefully selects solutions for its new neighborhood (N*(x)).
- TS Employs memory structures called the "tabu list" to filter eligible solutions.

What is Tabu Search?

- The tabu list consists of recently visited solutions. It typically includes solutions within the last 'n' iterations.
- 'n' represents the tabu tenure.
- In some cases, the list also includes solutions resulting from transitions between states.
- The tabu list ensures comprehensive exploration of the solution space.
- Balancing between exploiting known solutions and venturing into new areas.

Tabu Search: a high-level pseudo-code

procedure Tabu_Search()
  Define a neighborhood function $\mathcal{N}$ and select a local search algorithm *localSearch()* for $\mathcal{N}$;
  $t := 0$;
  $s_t :=$ Generate a starting solution; // e.g., with a construction heuristic
  $s^{best} := s_t$;
  $\mathcal{H}_t := (s_t,\ s^{best}, t)$; // initialize the memory data structure
  while ($\neg$ termination_criteria$(s_t, \mathcal{H}_t)$)
    $t := t + 1$;
    Depending on $(s_t, \mathcal{H}_t)$, generate a subset $N_t^{\mathcal{H}}(s_t) \subseteq \mathcal{N}(s_t)$;
    $s_t :=$ Best $\{s \mid s \in N_t^{\mathcal{H}}(s_t)\}$ with respect to f or some $f_t^{\mathcal{H}} = F(f, s_t, \mathcal{H}_t)$;
    if $(f(s_t) < f(s^{best}))$
      $s^{best} := s_t$;
    Update the memory data structure $\mathcal{H}_t$;
  end while
return s^{best};

Tabu Search Family

• Local search algorithms are a family of optimization techniques that iteratively explore the neighborhood of a current solution to find an improved solution.

• These algorithms are used to solve combinatorial optimization problems where the goal is to find the best solution among a finite set of possibilities.

Mimicking Human Decision-Making

• Tabu Search is an optimization technique inspired by the observation of human decision-making and natural processes.

• Humans often use intuition and experience to make decisions and avoid repeating past mistakes.

• Tabu Search's "tabu list" emulates this human-like learning by avoiding revisiting recently explored solutions.

Emulating Natural Processes

• Tabu Search also borrows inspiration from natural ecosystems.

• In ecosystems, species adapt and avoid harmful environments to survive and thrive.

• Tabu Search mimics this by using memory structures to adapt and avoid revisiting suboptimal solutions, promoting diversity in the search process.

Tabu Search Components

In this part, we will explore Tabu Search components in detail to understand how they collectively guide the search process and lead to improved results. Let's delve into the key elements of Tabu Search and how they contribute to solving optimization challenges.

Initial Solution

The initial solution for a Tabu Search metaheuristic depends on the specific problem you are trying to solve.

In many cases, the initial solution for Tabu Search can be generated using a simple heuristic or a random process, such as:

Random Solution: You can generate an initial solution randomly. This approach is often used when there is no better way to create an initial solution.

Greedy Solution: You can use a greedy algorithm to construct an initial solution. This means making a series of locally optimal choices at each step to build the initial solution. Greedy solutions are often better than random ones but may not be globally optimal.

Nearest Neighbor: In some problems, like the Traveling Salesman Problem (TSP), you can start with a nearest neighbor solution. Begin with an arbitrary point and repeatedly choose the nearest unvisited point until you've visited all points.

Constructive Heuristics: Depending on the nature of the problem, you might have specific heuristics that can be used to construct a reasonable initial solution.

Move function

The move function, often referred to as the neighborhood structure or the move generation mechanism, defines how new solutions are generated from the current solution. The move function plays a crucial role in exploring the solution space and finding better solutions. It is a mechanism which can obtain a new set of neighbor solutions by applying a small perturbation to a given solution. Each neighbor solution is reached immediately from a given solution by a move. Neighborhood structure is directly effective on the efficiency of TS algorithm. Therefore, unnecessary and infeasible moves must be eliminated if it is possible.

Tabu list and tabu tenure

Tabu list and Tabu tenure are key components used to manage and guide the search process. They are part of the strategy to prevent the algorithm from revisiting certain solutions or moves in order to promote diversification and escape local optima.

Tabu List: is a data structure that keeps track of recently visited solutions or moves. It stores information about solutions or moves that are temporarily prohibited from being selected as the next step in the search process. The tabu list serves to promote diversification by preventing the algorithm from returning to the same solutions repeatedly. It helps explore a broader region of the solution space.

Tabu Tenure: is a parameter that determines how long a solution or move remains on the tabu list. In other words, it defines the number of iterations for which a solution or move is considered tabu. The tabu tenure is problem-specific and can be set based on heuristics or experimentation. It can be a fixed value or adapt dynamically based on the problem and the algorithm's progress.

Aspiration criterion: the TS conditions at times prevent moves leading to unvisited solutions. Aspiration criterion is a condition that can override the tabu status of a certain move. To avoid certain missing solutions during the search.

The tabu list and tabu tenure work together as follows:

- When a solution is selected as the next step in the search process, it is added to the tabu list with a specified tabu tenure.

- While a solution or move is on the tabu list, it cannot be chosen as the next step in the search, even if it appears to be the best option according to the objective function.

-An exception to this rule is made when an aspiration criterion is met. If a solution that would typically be considered tabu offers a significant improvement over the current solution, and it satisfies the aspiration criteria, it may be allowed to be explored, even if it is technically tabu.

- After the tabu tenure expires for a particular solution or move, it is removed from the tabu list and can be reconsidered as a possible choice in subsequent iterations.

Stopping Criteria

Stopping Criteria determine when the algorithm should terminate its search. The choice of stopping criteria can have a significant impact on the quality of the solution found and the computational resources used.

Maximum Iterations: You can set a maximum number of iterations, and the Tabu Search algorithm stops once this limit is reached. This is a simple and widely used stopping criterion, especially when the algorithm is applied to problems where the optimal solution is not known.

Time Limit: Instead of specifying a fixed number of iterations, you can set a time limit for the algorithm. The algorithm will terminate once the specified amount of time has passed, which can be useful in situations where you want to control the computational resources dedicated to the search.

Target Solution: If you have a specific target solution quality in mind, you can set a stopping criterion based on achieving that solution quality or a certain percentage of the optimal solution quality.

Convergence: You can monitor the convergence of the algorithm by tracking the best solution quality over time. If it stabilizes or shows little improvement, you may choose to stop the algorithm.

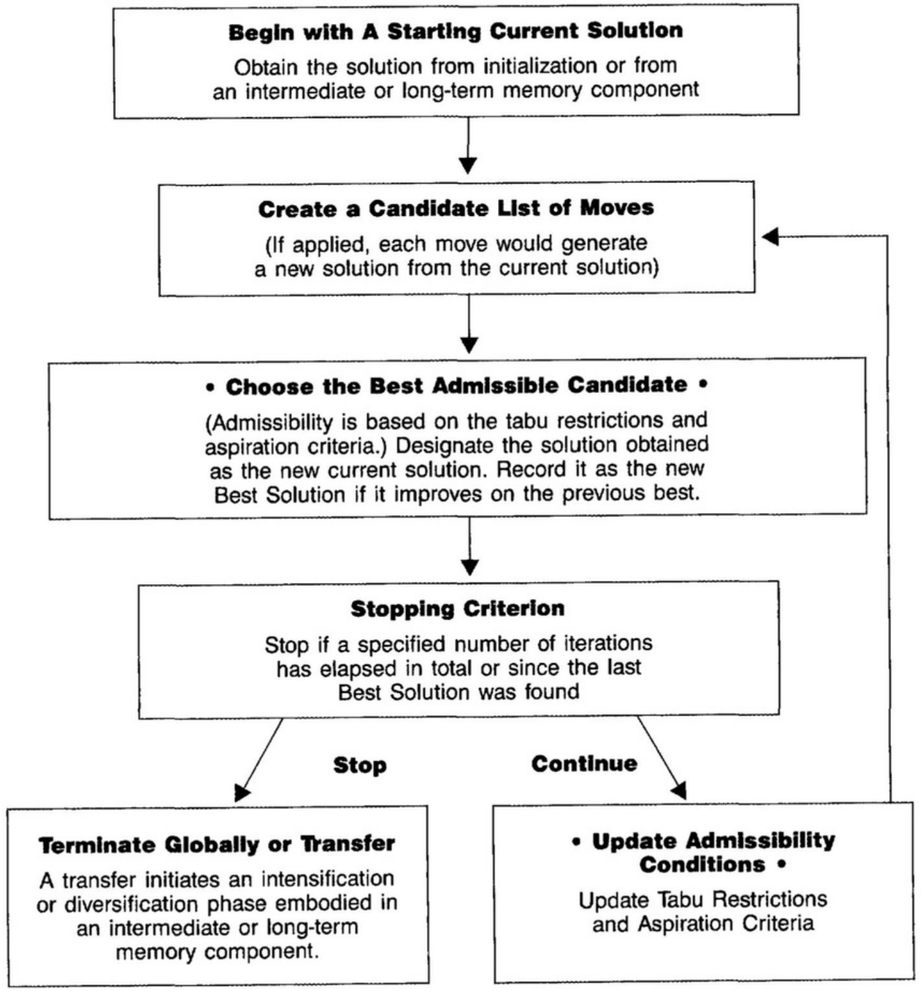


Fig 1: Tabu Search Short-Term Memory Component

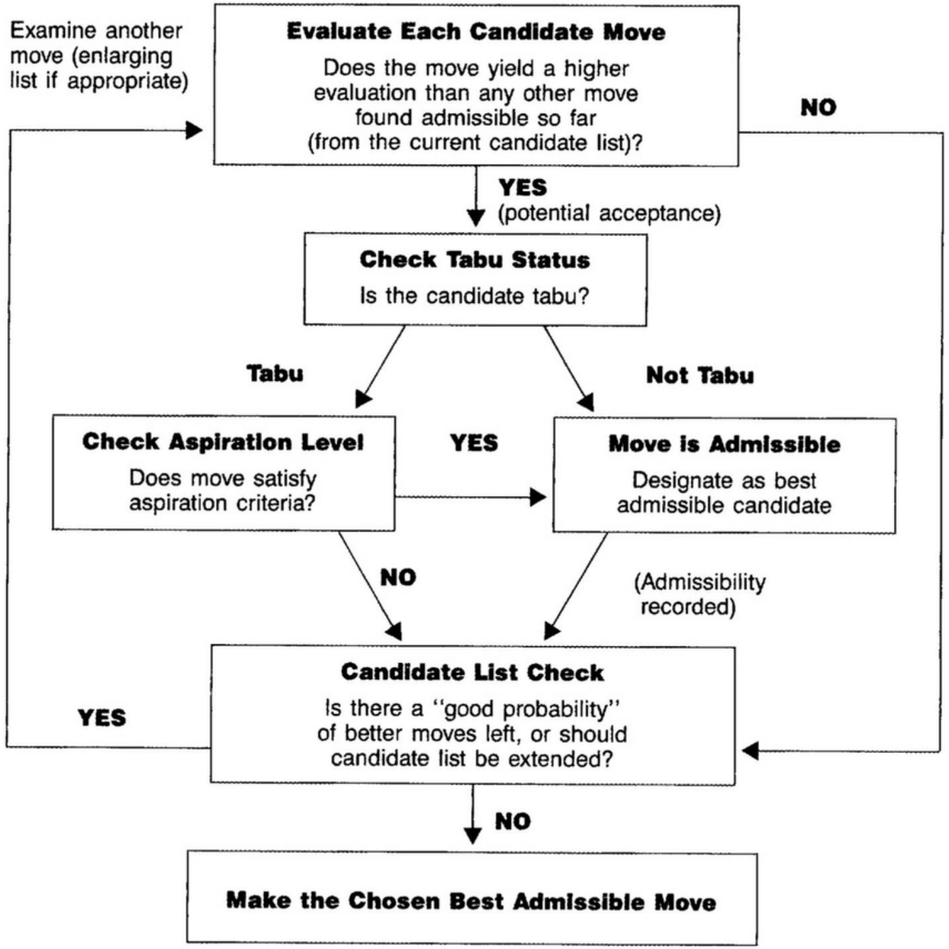


Fig 2: Selecting the best admissible candidate

Intensification

Intensification in the context of optimization is the process of concentrating search efforts on promising regions within the solution space.
It tries to direct the search algorithm towards the areas where optimal solutions are likely to be found.

Intensification with tabu search

Local Search: Intensification in tabu search often involves making small adjustments to the current solution in order to improve it by exploring nearby solutions, using operations like swapping, perturbing, or modifying elements.

Escaping Plateaus: Intensification is crucial when the search process hits plateaus, areas where it can stall. These plateaus may not necessarily be the global or local optima, but they can impede the search progress. Intensification helps the algorithm escape these plateaus by concentrating search efforts in nearby regions that have the potential for improvement.

Reducing Search Space: As the optimization process proceeds, and better solutions are discovered through intensification, the algorithm narrows down the areas to be explored. This reduction in the search space is advantageous because it helps the algorithm converge more quickly to an optimal solution.

Diversification

Diversification in optimization involves maintaining or increasing the variety of explored solutions to prevent the search from getting stuck in a limited area of the solution space. It's a strategy used to ensure a broader range of possible solutions is considered,

Diversification with tabu search

Exploring Varied Promising Areas: Diversification is the process of actively seeking out different and promising parts within the search space, making sure you don't focus too narrowly.

Primarily Random-Based: Diversification often relies on randomness in its approach, introducing elements of chance to discover new areas rather than following a strict, predetermined path.

Diversification with tabu search

Utilizing Memory: It can also employ memory to keep track of where it has been before. This helps prevent revisiting the same places and instead guides the exploration towards untapped regions.

Not Always About Improvement: Diversification doesn't always aim to make the current solution better. Its primary goal is to ensure that the search remains broad, rather than focusing solely on optimizing the current solution.

Combining Intensification and Diversification

Balancing Local and Global Search: intensification, which hones the current solution using local search, and **diversification**, which explores different solution regions.

Preventing Premature Convergence: Diversification prevents premature convergence to suboptimal solutions, particularly when the initial solution is far from the global optimum. It keeps the search wide, delaying convergence and improving the chances of finding better solutions. **Intensification** then refines solutions in the promising areas identified during diversification.

Combining Intensification and Diversification

Improved Solution Quality: Intensification and **diversification** together result in finding higher-quality solutions. **Intensification** improves nearby solutions, while **diversification** keeps exploration fresh. This balanced approach helps identify optimal or near-optimal solutions, even in complex optimization scenarios.

Adaptive Strategy: Tabu Search can dynamically adjust the emphasis on **intensification** and **diversification** as the search evolves. Initially, it focuses on **diversification** to explore broadly, shifting towards **intensification** as promising areas emerge.

Techniques used for diversificationin Tabu Search.

- Random Restart
- Tabu Tenure Adjustment
- Solution Perturbation
- Path Diversification:

Random Restart

- **Random restart** is a simple diversification technique where the algorithm periodically restarts the search from a random or different initial solution. This helps to escape local optima and explore different parts of the solution space.

- **Practical Application:** Random restart is commonly used in optimization problems with multiple local optima, such as the N-Queens problem. It ensures that the algorithm explores different initial configurations of the chessboard to avoid getting stuck in local optima.

Tabu Tenure Adjustment:

- **Adjusting tabu tenure** dynamically can be a diversification technique. By changing the tabu tenure for certain moves, the algorithm can explore different regions of the solution space more or less frequently.

- **Practical Application: Tabu tenure** adjustment is often used in job scheduling problems. By modifying the tabu tenure for specific job assignments or machine allocations, the algorithm can diversify its exploration strategies.

Solution Perturbation:

- **Solution Perturbation** involves introducing small random changes to the current solution. These changes can include swapping elements, altering variables, or perturbing the solution in a controlled manner. Perturbation introduces diversity into the search process.

- **Practical Application: Solution perturbation** is commonly used in optimization problems like the Traveling Salesman Problem (TSP). By perturbing the order of cities in a tour, the algorithm can explore alternative routes.

Path Diversification

- **Path Diversification** involves exploring alternative paths or routes in the solution space. Instead of focusing on a single path, the algorithm considers multiple paths in parallel or as part of the diversification strategy.

- **Practical Application: Path Diversification** is used in vehicle routing problems to explore different routes that can lead to cost savings. It allows the algorithm to consider various routes for delivering goods.

Memory structures

- Short-Term Memory
- Long-Term Memory

Short-Term Memory

- **Short-Term Memory** refers to the memory structure that stores information about recently visited solutions or moves. It helps prevent the algorithm from revisiting the same solutions or making the same moves in the immediate future.

- **It is primarily associated with intensification.**

Long-Term Memory

- **Long-Term Memory** is a memory structure that stores information about the best solutions encountered throughout the entire search process. It ensures that the algorithm remembers the globally or locally optimal solutions found so far.

- **It primarily supports diversification.**

Real World Applications using Tabu Search

- **Combinatorial Optimization** (Traveling Salesman Problem, Job Shop Scheduling Problem, Vehicle Routing Problem,...)
- **Engineering and Manufacturing** (Production Scheduling, Quality Control, Supply Chain Management,...)
- **Telecommunications** (Wireless Network Design, Frequency Assignment, Routing and Network Design,...)

Implementation of Tabu Search in Python

Conclusion

In conclusion, we have explored the concept of Tabu Search, its roots in the field of optimization, and its key components, including intensification and diversification strategies. We have also discussed real-world applications where Tabu Search has proven to be a valuable problem-solving tool. Furthermore, we've seen how this powerful technique can be practically implemented in Python. Tabu Search offers a versatile and effective approach to solving complex optimization problems, making it a valuable asset for researchers and practitioners in various domains.

References

https://www.sciencedirect.com/science/article/pii/S030505480500398 9

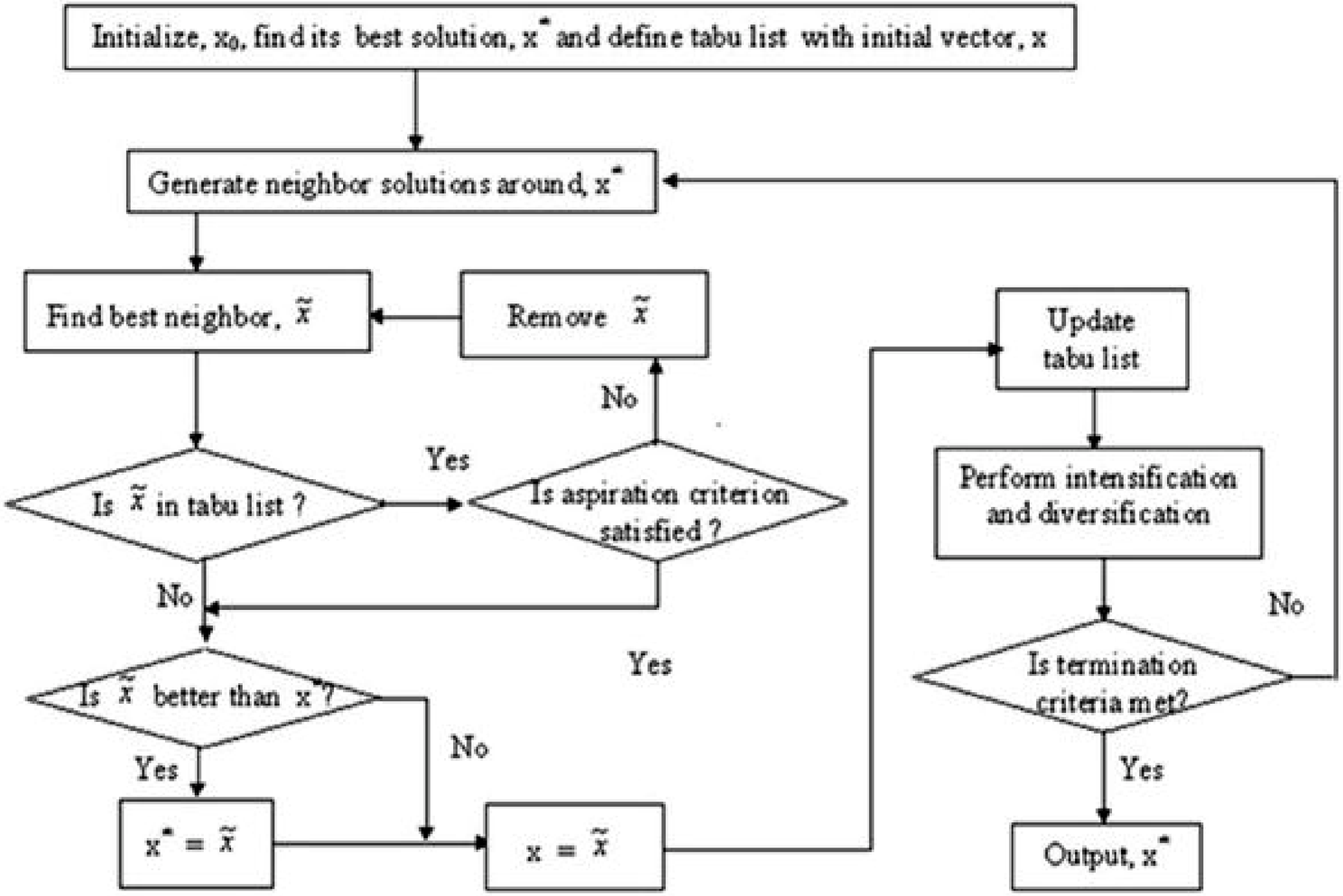
https://www.sciencedirect.com/science/article/pii/S009813540300276X

http://old.math.nsc.ru/LBRT/k5/OR-MMF/2019_Book_HandbookOfMetaheuristics.pdf

https://www.intechopen.com/chapters/77046

https://medium.com/@m.kaleia/tabu-search-gentle-introduction-46c479eb6525

https://www.researchgate.net/publication/242527226_Tabu_Search_A_Tutorial


Initialize, x_0, find its best solution, x^* and define tabu list with initial vector, x
Generate neighbor solutions around, x^*
Find best neighbor, $\tilde{x}$
Remove $\tilde{x}$
No
Yes
Is $\tilde{x}$ in tabu list ?
Is aspiration criterion satisfied ?
No
Yes
Is $\tilde{x}$ better than x^*?
No
Yes
$x^* = \tilde{x}$
$x = \tilde{x}$
Update tabu list
Perform intensification and diversification
No
Is termination criteria met?
Yes
Output, x^*