Chapter 4: Tabu Search Algorithm

CS-616: Optimization Algorithms

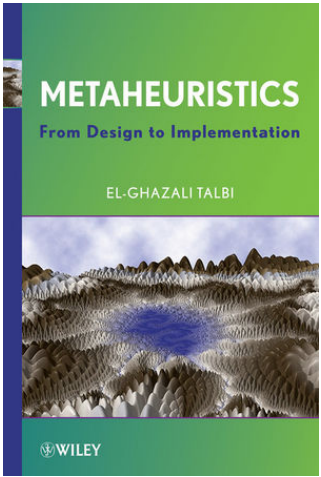
Dr. Mahdi A. Khemakhem & Dr. Esraa M. Eldesouky

Department of Computer Science
College of Computer Engineering and Science
Prince Sattam bin Abdulaziz University

AY: 2025/2026

Materials

- **Textbook:** **"Metaheuristics: From Design to Implementation"** by *El-Ghazali Talbi*
- ➥Read Chapter 2 (Section 2.5) for this chapter's material



Outline

Outline

History

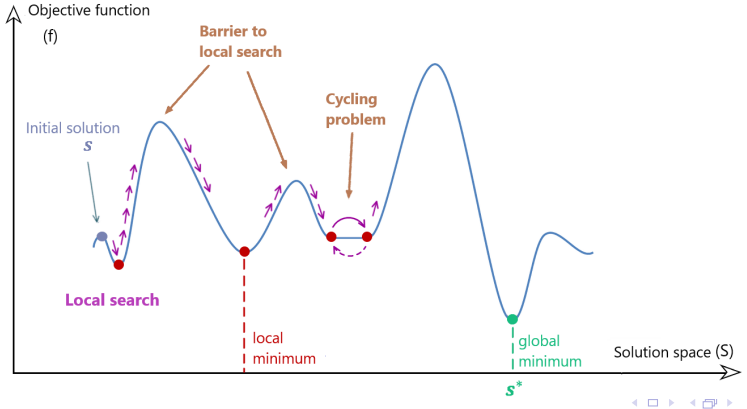
- Tabu Search (TS) algorithm was proposed by Fred Glover in 1989.
- In a parallel work, a similar approach named Steepest Ascent/Mildest Descent (SA/MD) has been proposed by Pierre Hansen.
- In the 1990s, the TS algorithm became very popular in solving optimization problems in an approximate manner.
- Nowadays, it is one of the most widespread S-Metaheuristics.
- The use of memory, which stores information related to the search process, represents the particular feature of TS.

Fred Glover

Pierre Hansen

Tabu Search Principle (1/2)

Tabu Search behaves like a steepest Local Search algorithm, but it accepts non-improving solutions to escape from local optima when all neighbors are non-improving solutions.



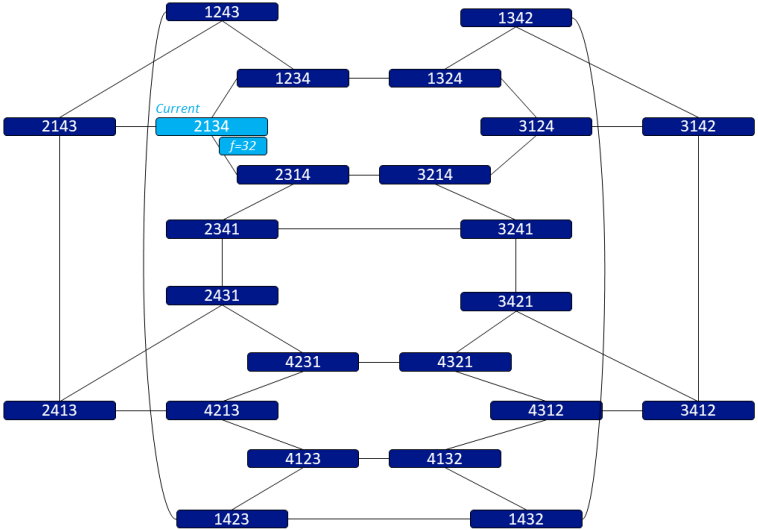
Tabu Search Principle (2/2)

- Usually, TS explores the whole neighborhood in a deterministic manner.
- As in LS, when a better neighbor is found, TS replaces the current solution.
- When a local optima is reached, the TS searching process carries on by selecting a candidate worse than the current solution.
  - The best solution in the neighborhood is selected as the new current solution even if it is not improving the current solution.
  - This policy may generate cycles; that is, previous visited solutions could be selected again.
  - Cycles should be avoided using a memory that saves all or a part of previous selected solution.

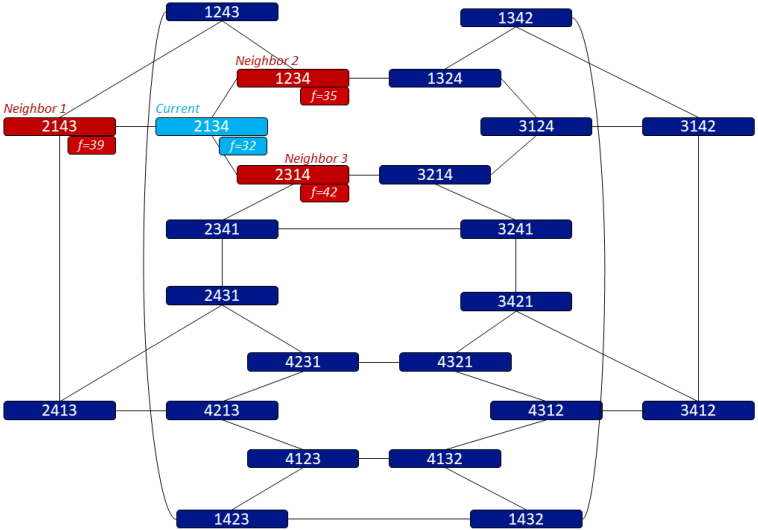
Avoid Cycles by Tabu List (1/2)

- To avoid cycles, TS discards the neighbors that have been previously visited. It memorizes the recent search trajectory.
- Tabu Search manages a memory of the solutions or moves recently applied, which is called the Tabu List.
  - This Tabu List constitutes the short-term memory.
  - At each iteration of TS, the Short-Term memory is updated.
  - Storing all visited solutions is time and space consuming. Indeed, we have to check at each iteration if a generated solution does not belong to the list of all visited solutions.
  - Usually, the attributes of the moves are stored in the Tabu List.

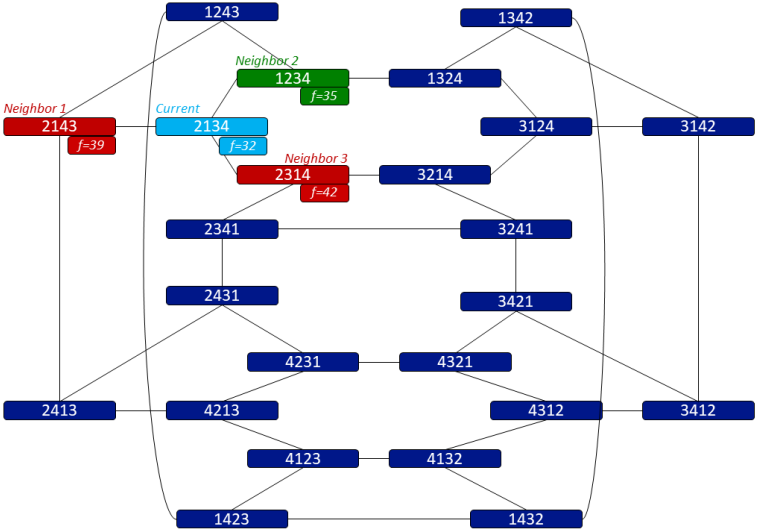
Avoid Cycles by Tabu List (2/2)



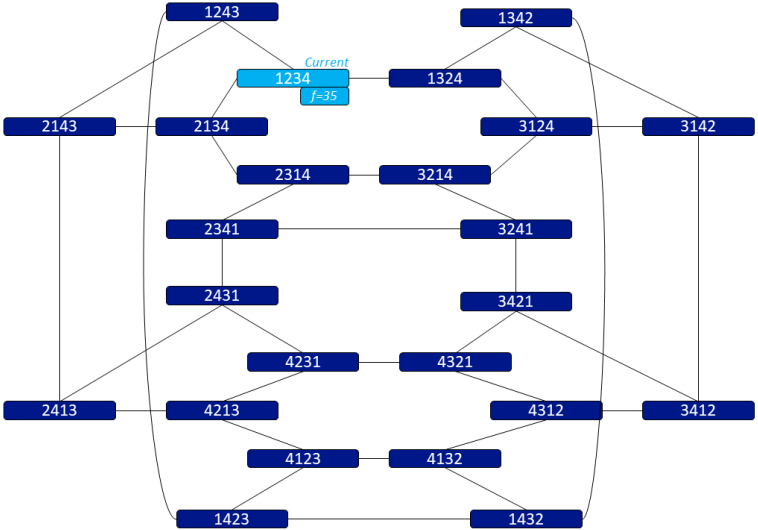
Avoid Cycles by Tabu List (2/2)



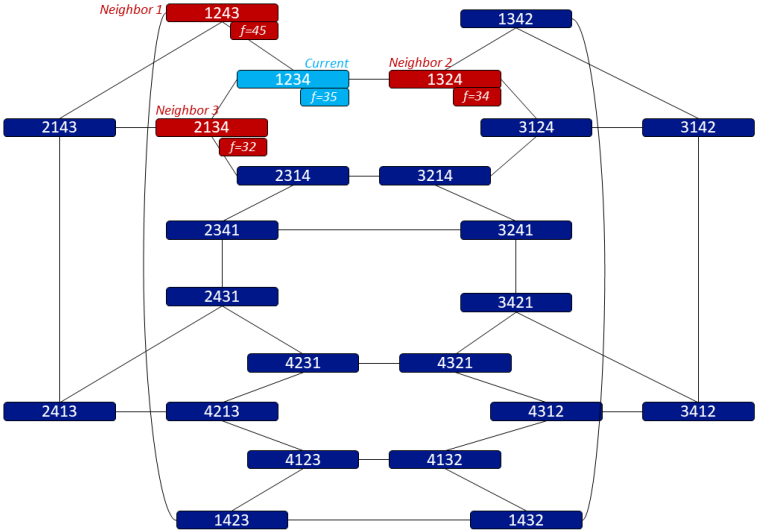
Avoid Cycles by Tabu List (2/2)



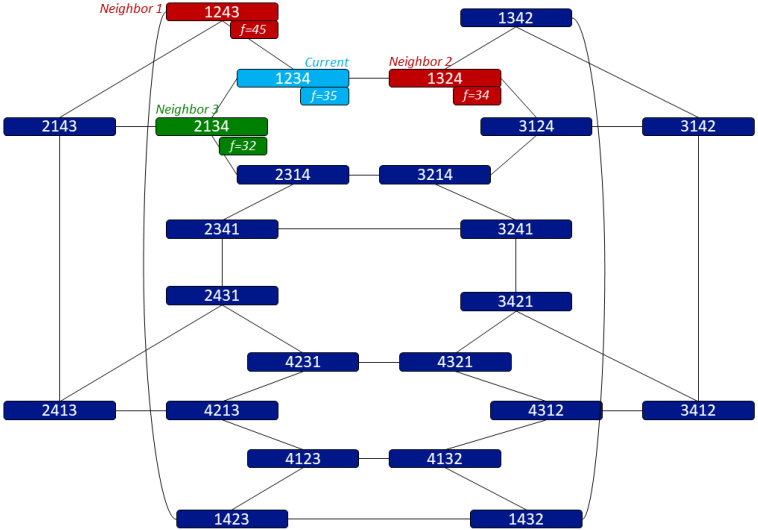
Avoid Cycles by Tabu List (2/2)



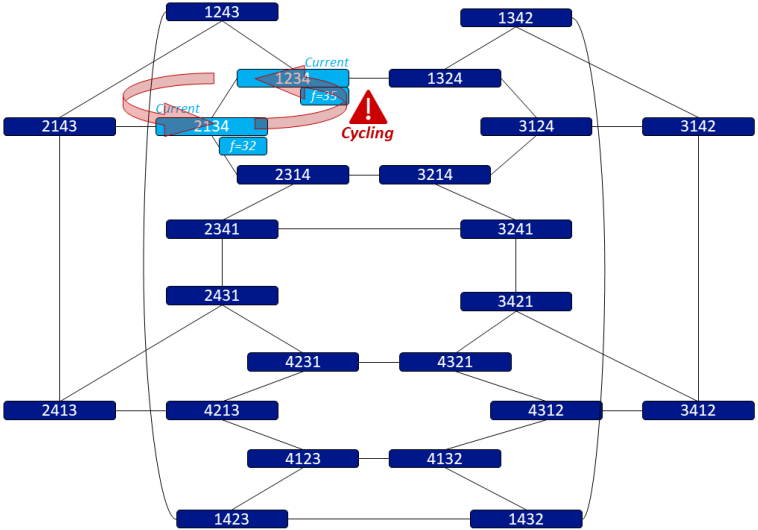
Avoid Cycles by Tabu List (2/2)



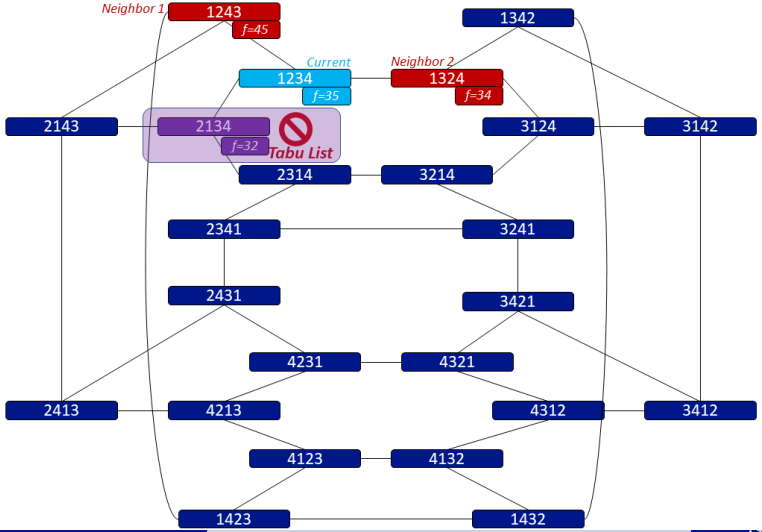
Avoid Cycles by Tabu List (2/2)



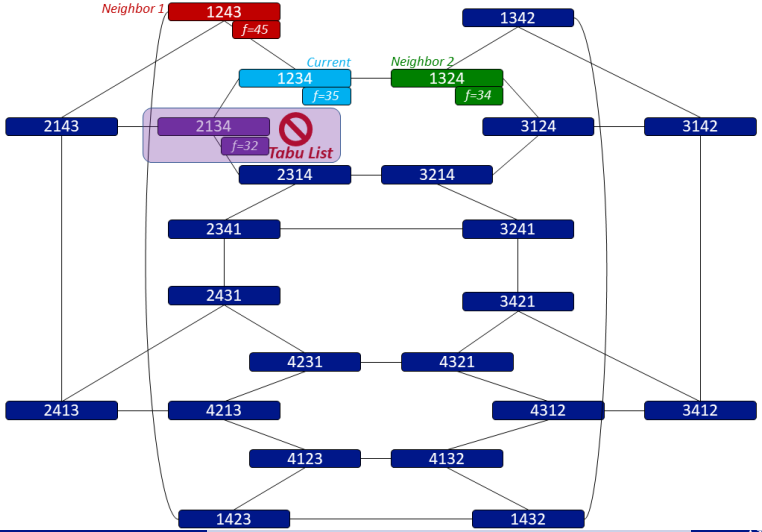
Avoid Cycles by Tabu List (2/2)



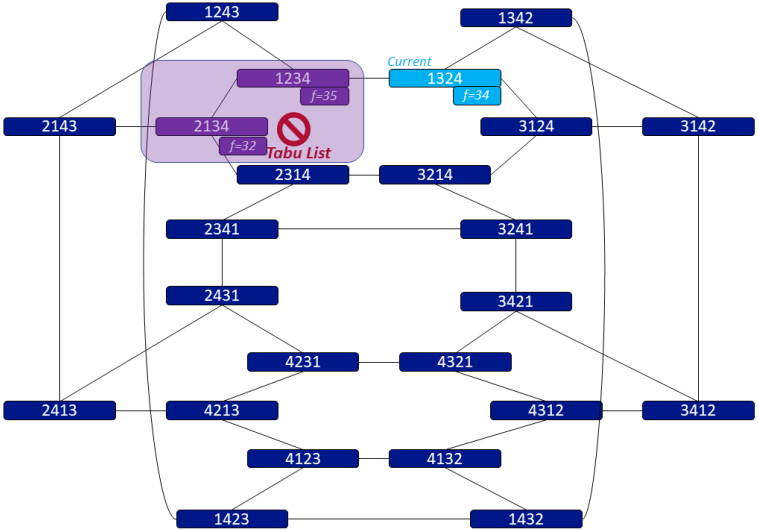
Avoid Cycles by Tabu List (2/2)



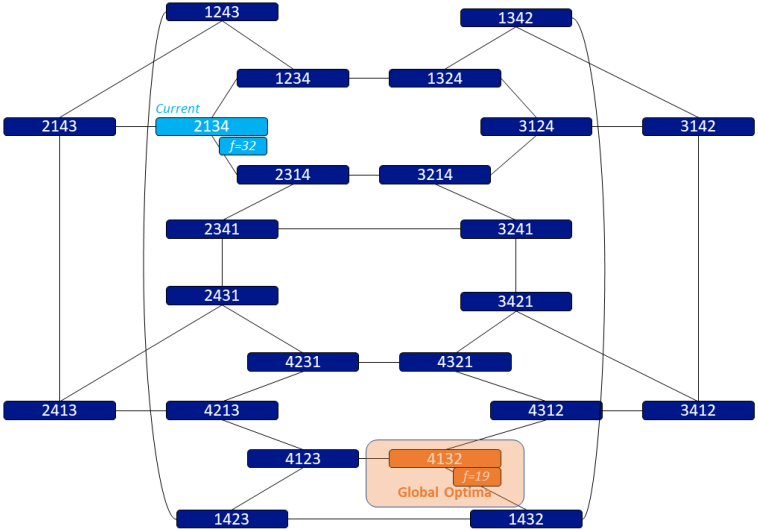
Avoid Cycles by Tabu List (2/2)



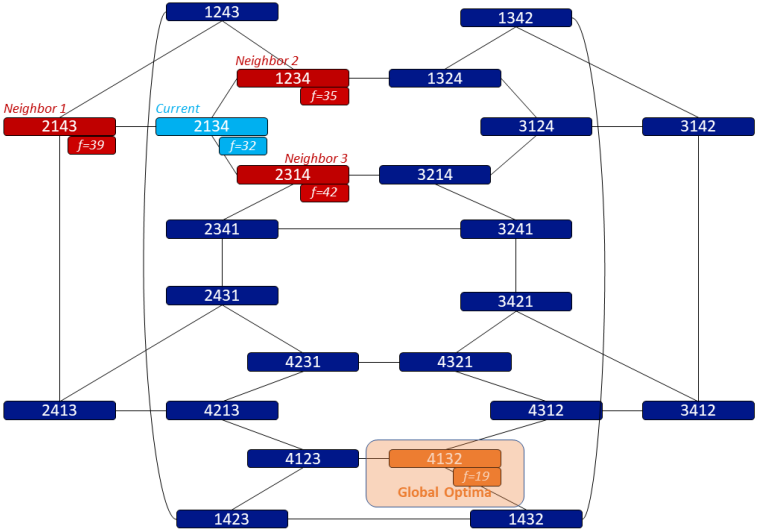
Avoid Cycles by Tabu List (2/2)



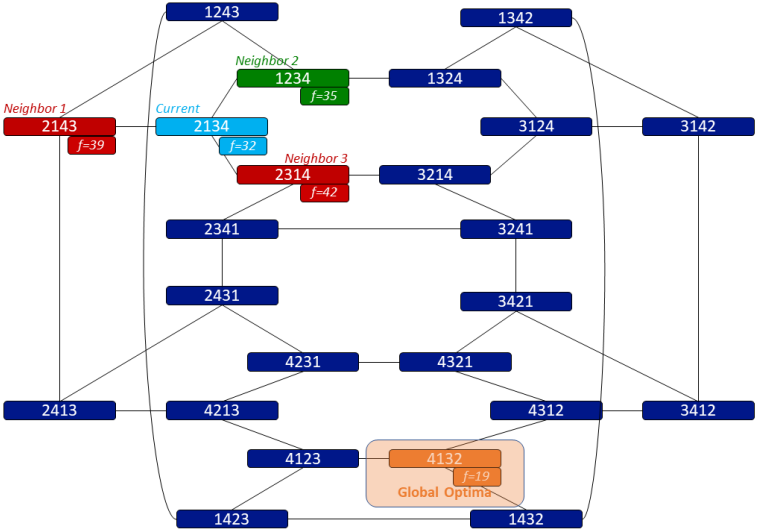
Tabu Search Process



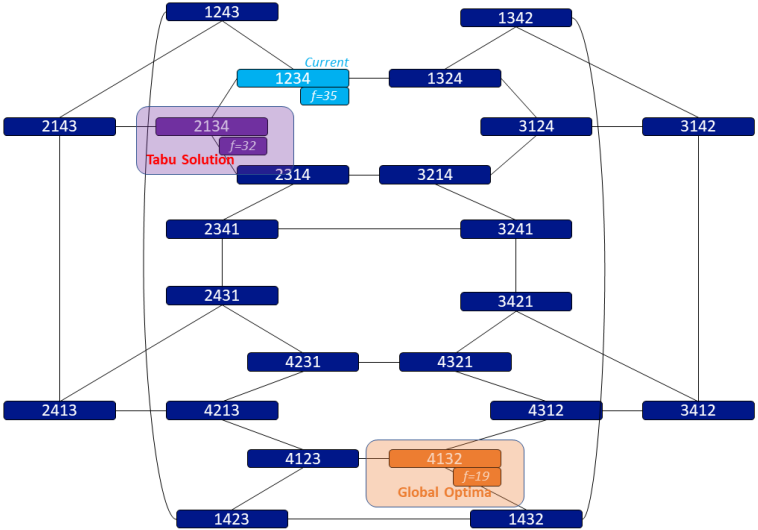
Tabu Search Process



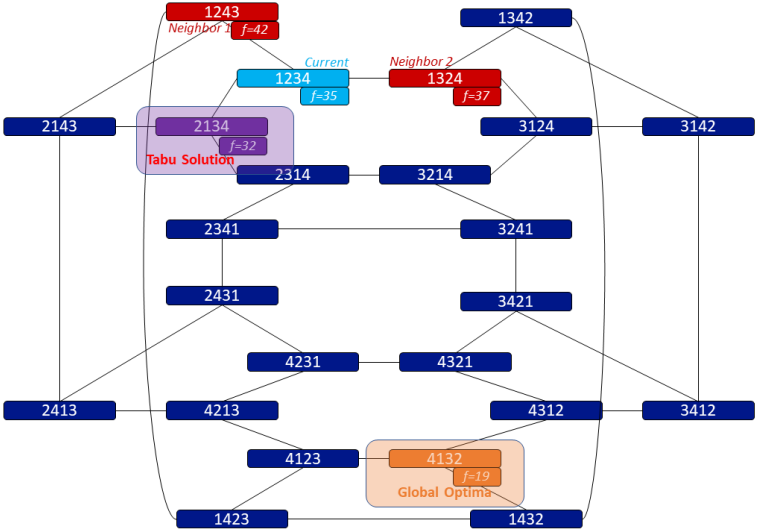
Tabu Search Process



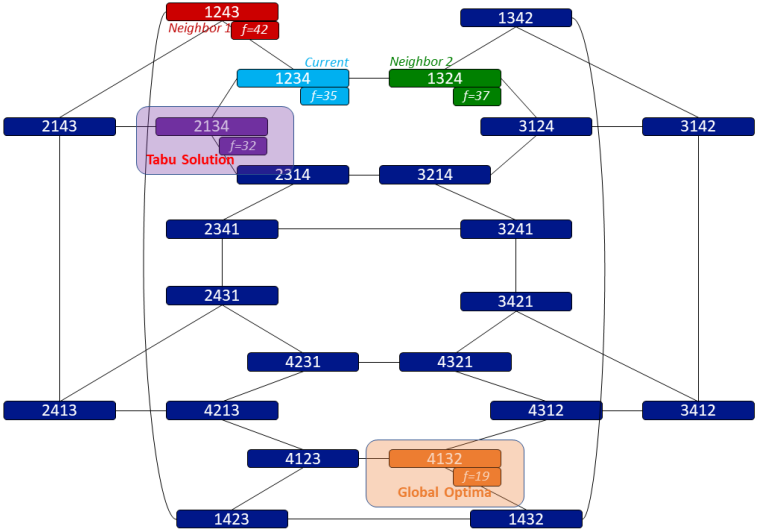
Tabu Search Process



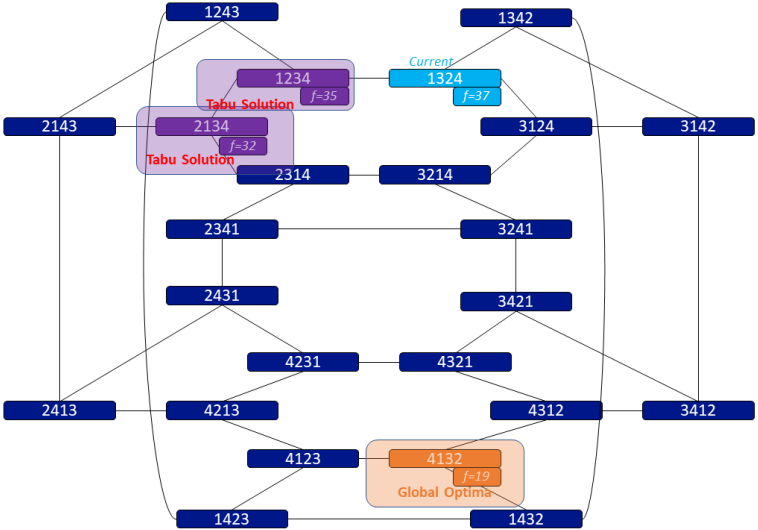
Tabu Search Process



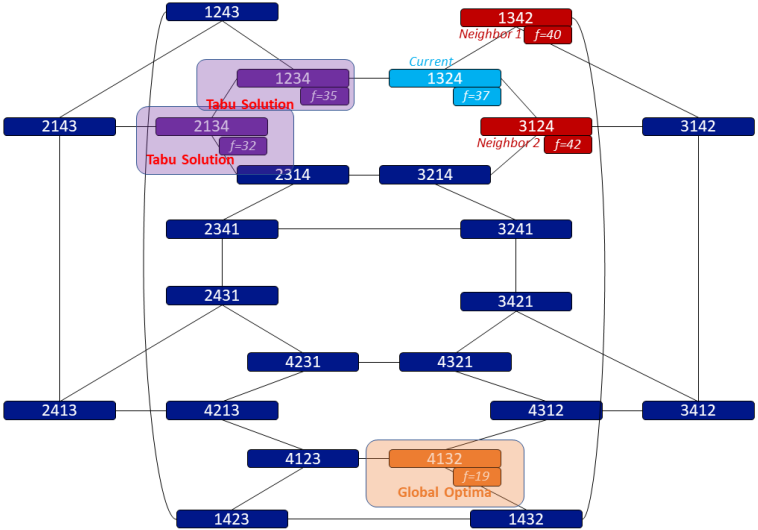
Tabu Search Process



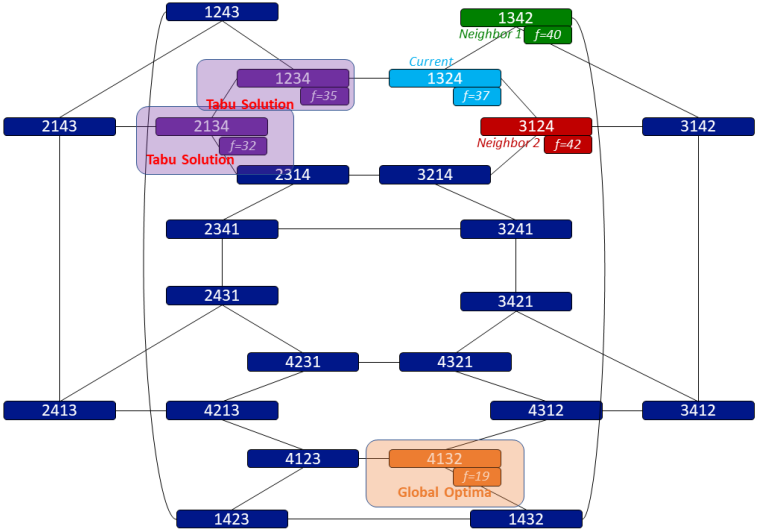
Tabu Search Process



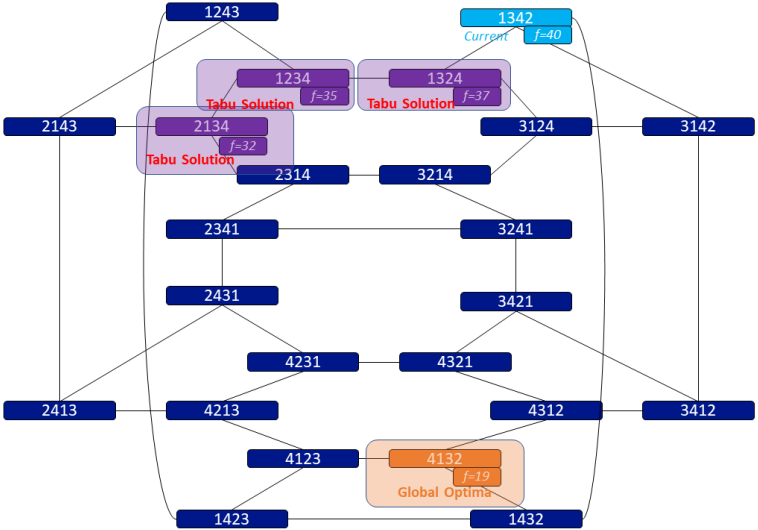
Tabu Search Process



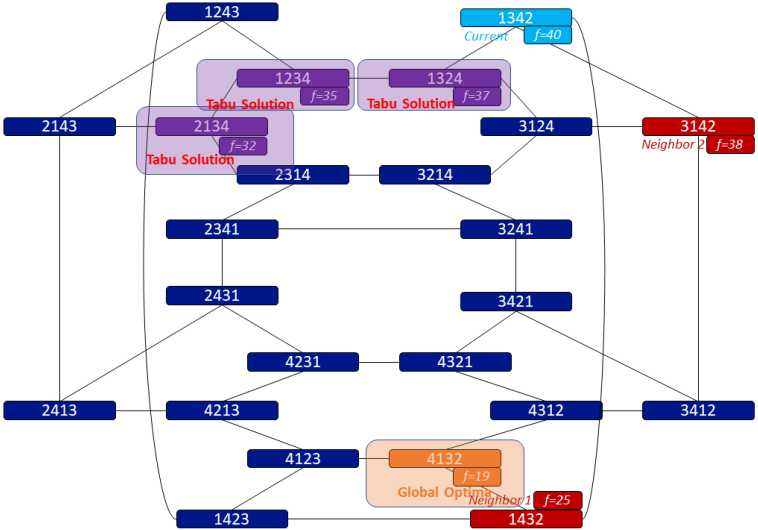
Tabu Search Process



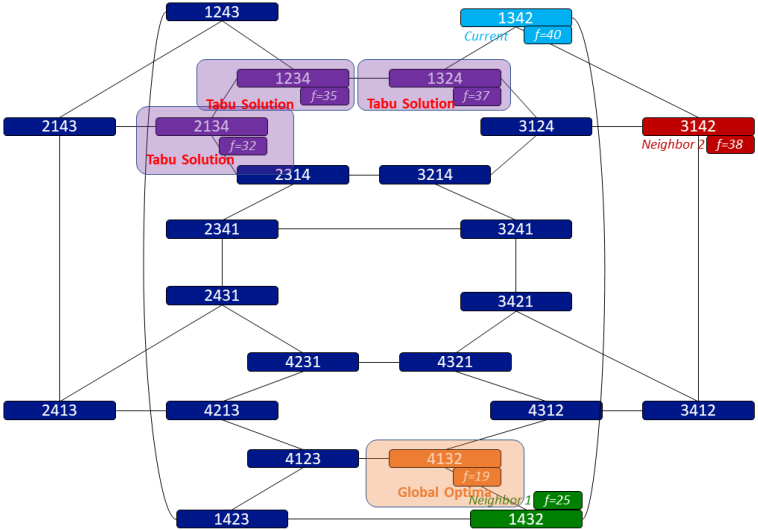
Tabu Search Process



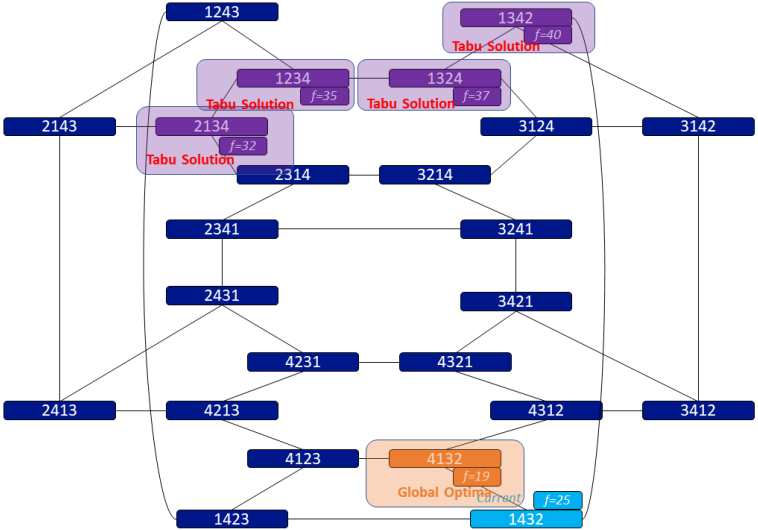
Tabu Search Process



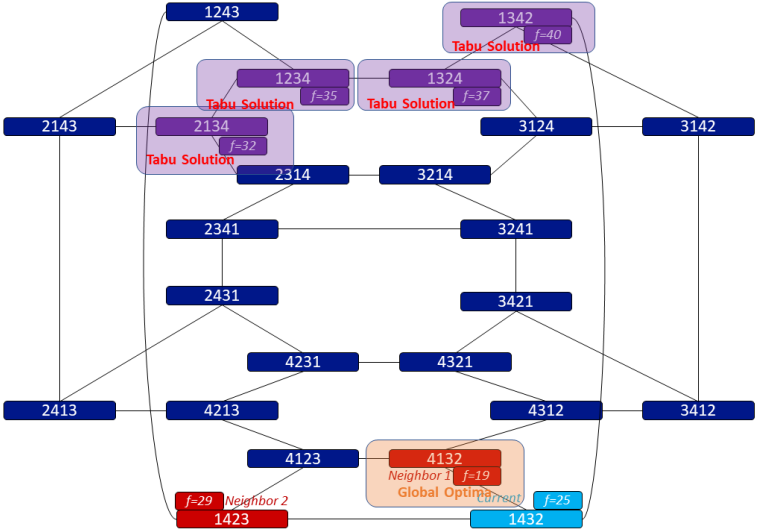
Tabu Search Process



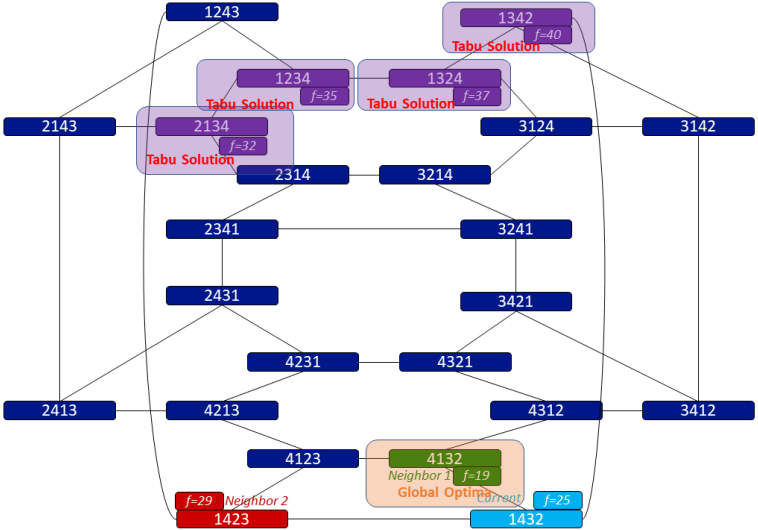
Tabu Search Process



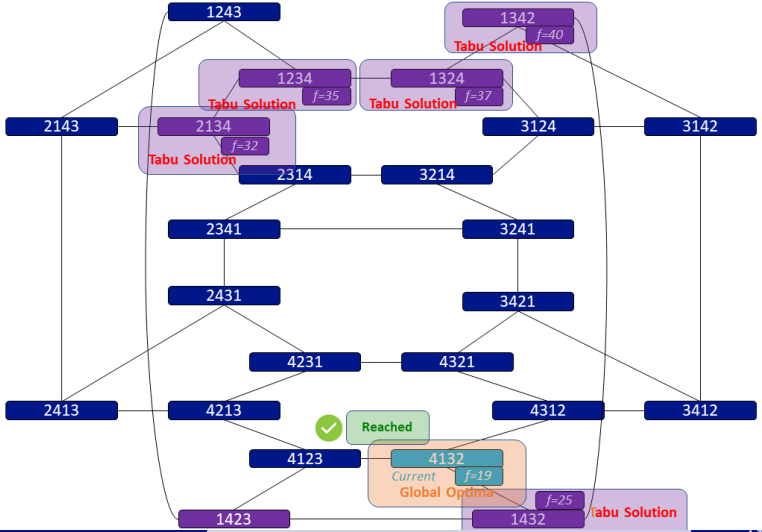
Tabu Search Process



Tabu Search Process



Tabu Search Process



Outline

Aspiration Criteria to Deal with the Tabu List Restriction!

- By introducing the concept of solution features or move features in the Tabu List, TS may lose some information about the search memory.
  - TS can reject solutions that have not yet been generated.
  - If a move is "good", but it is Tabu, do we still reject it?
- The Tabu List may be too restrictive; a non-generated solution may be forbidden.
  - Yet for some conditions, called **Aspiration Criteria**, Tabu solutions may be accepted.
- The admissible neighbor solutions are those that are non-Tabu or hold the **Aspiration Criteria**.

Tabu List/Aspiration Criteria

In addition to the common design issues for S-Metaheuristics such as the definition of the neighborhood and the generation of the initial solution, the main design issues that are specific to a simple TS are:

- **Tabu List (Short-Term memory):** The goal of using the Short-Term memory is to prevent the search from revisiting previously visited solutions. As mentioned, storing the list of all visited solutions is not practical for efficiency issues.
- **Aspiration Criterion:** A commonly used Aspiration Criterion consists in selecting a Tabu move if it generates a solution that is better than the best found solution. Another aspiration criterion may be a Tabu move that yields a better solution among the set of solutions possessing a given attribute.

Intensification/Diversification

Some advanced mechanisms are commonly introduced in Tabu Search to deal with the intensification and the diversification of the search:

- **Intensification (Intermediate-Term memory):** The Intermediate-Term memory stores the elite (e.g., best) solutions found during the search. The idea is to give priority to attributes of the set of elite solutions, usually in weighted probability manner.
- **Diversification (Long-Term memory):** The Long-Term memory stores information on the visited solutions along the search. It explores the unvisited areas of the solution space.
  For instance, it will discourage the attributes of elite solutions in the generated solutions to diversify the search to other areas of the search space.

Outline

Tabu Search - General Schema

Algorithm 1 shows the general schema of a Tabu Search Metaheuristic.

Algorithm 1: Template of the Tabu Search Algorithm

Input: Initial solution s_0; /* Initial solution generated by a Greedy Algorithm */
Output: Best solution found s^*; /* Local optima (Global optima is highly probable) */
Initialize the Tabu List, Intermediate-Term and Long-Term Memories;
$s = s^* = s_0$; /* s is the current solution, s^* is the best-known solution */
repeat
    Find best admissible neighbor $s' \in N(s)$; /* non Tabu or Aspiration Criterion holds */
    if $f(s') < f(s^*)$ **then**
        $s^* = s'$; /* Update the best known solution s^* if improved by s' */
    end
    $s = s'$;
    Update Tabu List, Aspiration conditions, Intermediate-Term and Long-Term memories;
    if *Intensification Criterion holds* **then**
        Apply Intensification;
    end
    if *Diversification Criterion holds* **then**
        Apply Diversification;
    end
until *Stopping criteria staisfied*;

Outline

Features of Tabu Search Memories

Three types of memories used by Tabu Search. The following table present the role and the popular representation(s) of each memory type.

Search Memory	Role	Popular Representation
Short-Term Memory (Tabu list)	Prevent cycling	Visited solutions, Moves attributes, Solution attributes
Intermediate-Term Memory	Intensification	Recency memory
Long-Term Memory	Diversification	Frequency memory

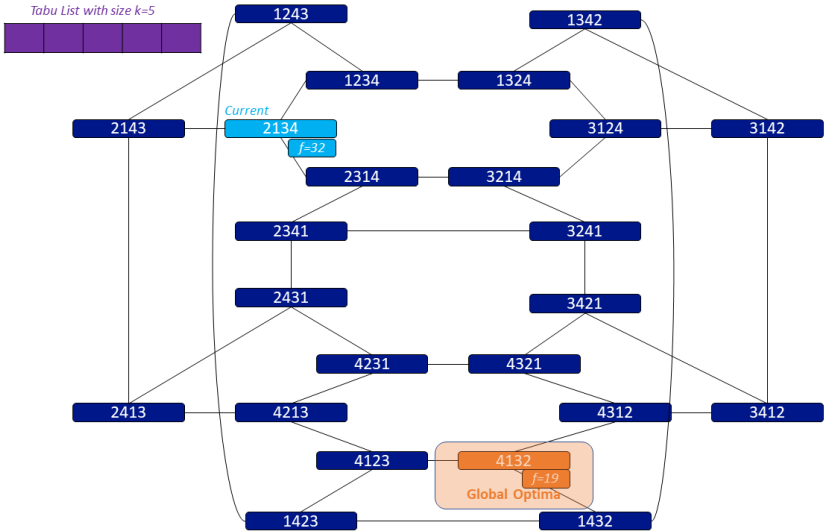
Short-Term Memory/Tabu List (1/2)

- The role of the Short-Term memory is to store the recent history of the search to prevent cycling.
- The naive (simple) straightforward representation consists in recording all visited solutions during the search.
  - Ensures the lack of cycles but is seldom used as it produces a high complexity of data storage and computational time.
- The first improvement to reduce the complexity of the algorithm is to limit the size of the Tabu List.
  - If the Tabu List contains the last k visited solutions, Tabu Search prevents a cycle of size at most k.

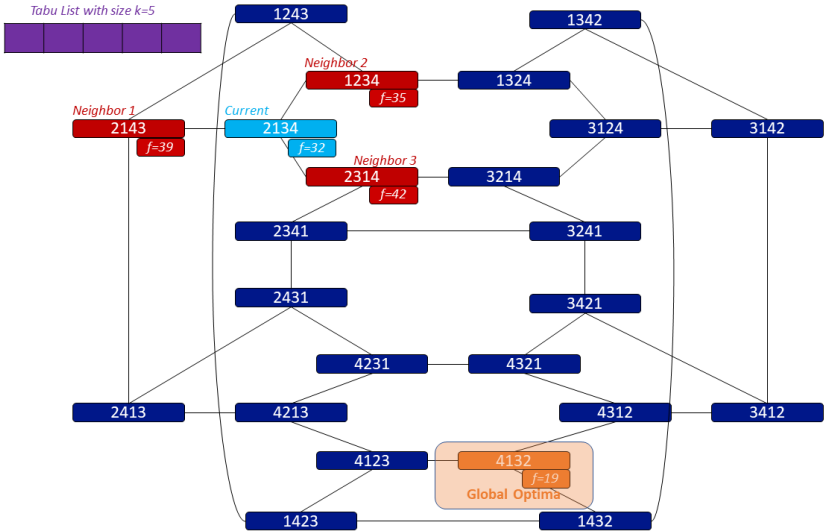
Short-Term Memory/Tabu List (2/2)

- In general, attributes of the solutions or moves are used.
  - Less important data storage and computational time but skips some information on the history of the search (e.g., the absence of cycles is not ensured).
- The most popular way to represent the Tabu List is to record the move attributes.
  - Tabu List composed of the reverse moves that are forbidden.
  - This scheme is directly related to the neighborhood structure being used to solve the problem.
  - If the move m is applied to the solution s_i to generate the solution s_j ($s_j = s_i \oplus m$), then the move m (or its reverse m^{-1} such that $(s_i \oplus m) \oplus m^{-1} = s_i$) is stored in the Tabu List.
  - This move is forbidden for a given number of iterations, named the **Tabu Tenure** of the move.
  - If the Tabu List contains the last k moves, only k cycles will be prevented.

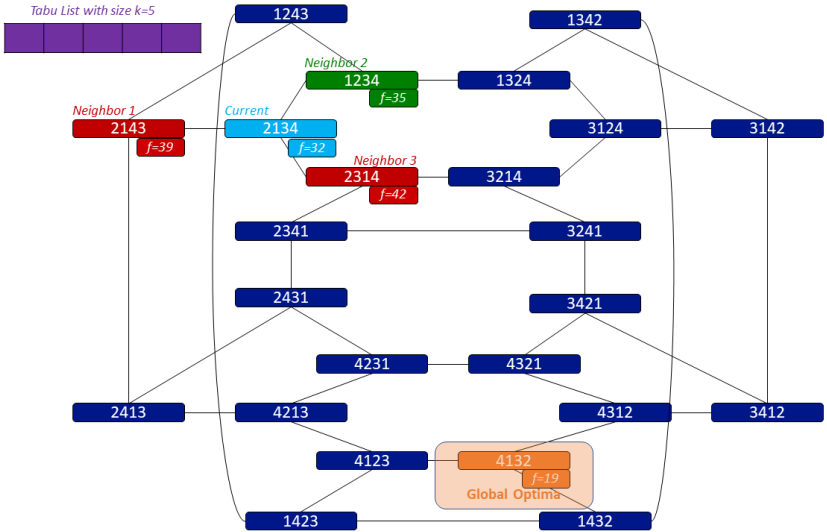
Example of Short-Term Memory



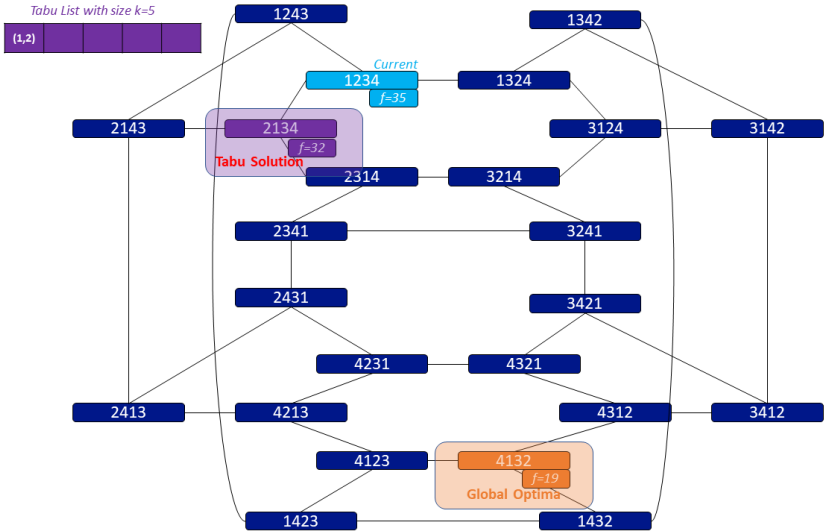
Example of Short-Term Memory



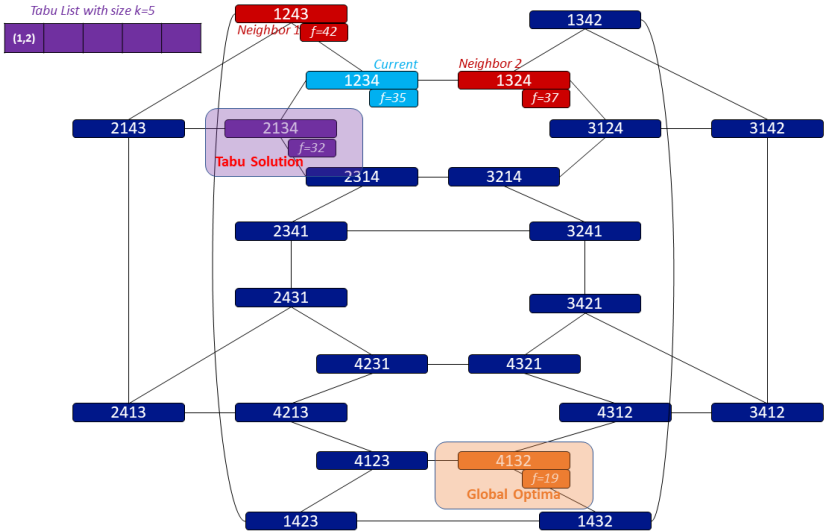
Example of Short-Term Memory



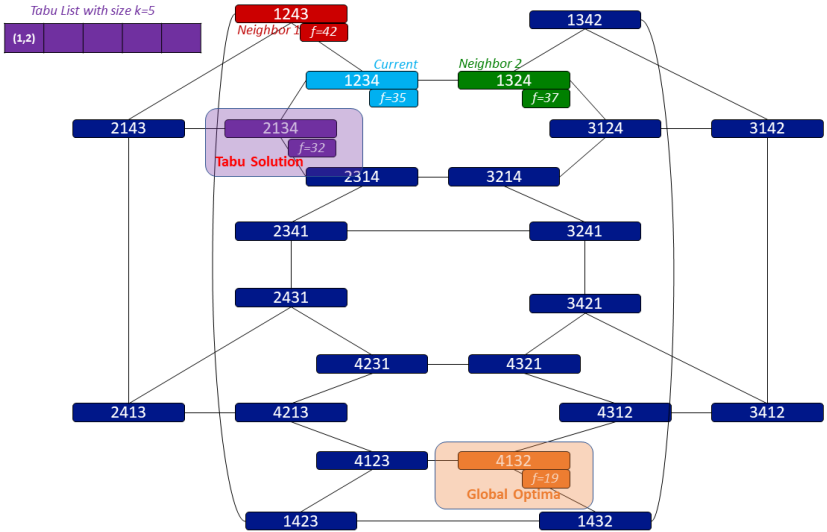
Example of Short-Term Memory



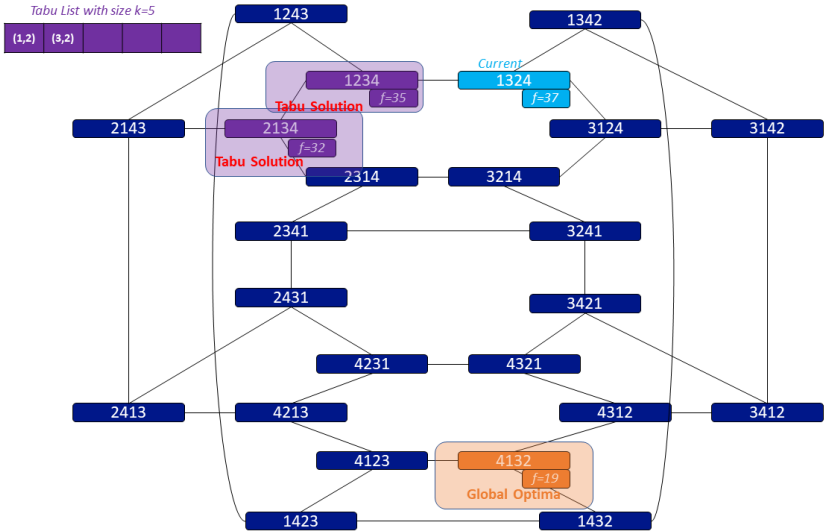
Example of Short-Term Memory



Example of Short-Term Memory



Example of Short-Term Memory



Size of Short-Term Memory k

- **The size of the Tabu List k is a critical parameter that has a great impact on the performances of the Tabu Search algorithm.**
  - **At each iteration, last move is added in the list, whereas oldest move is removed from the list.**
- **Smaller $k \Rightarrow$ higher probability of cycling.**
- **Larger $k \Rightarrow$ more restrictions $\Rightarrow$ encourage search diversification.**
- **A compromise must be found that depends on the landscape structure of the problem and its associated instances.**
- **The Tabu List size may take different forms:**
  - **Static:** static value is associated with the tabu list. It may depend on the problem instance size instance and particularly the neighborhood size.
  - **Dynamic:** may change during the search without using any information on the search memory.
  - **Adaptive:** updated according to the search memory.

Intermediate-Term Memory/Intensification (1/2)

The role of the intensification is to exploit the information of the best found solutions (Elite Solutions) to guide the search in promising regions of the search space.

- This information is stored in a Medium-Term Memory.
- The idea consists in extracting the (common) features of the Elite Solutions and then intensifying the search around solutions sharing those features.
- A popular approach consists in restarting the search with the best solution obtained and then fixing in this solution the most promising components extracted from the Elite Solutions.

Intermediate-Term Memory/Intensification (2/2)

The main representation used for the Medium-Term Memory is the Recency Memory[1].

- First, the components associated with a solution have to be defined; this is a problem specific task.
- The Recency Memory will memorize for each component the number of successive iterations the component is present in the visited solutions.
- The most commonly used event to start the intensification process is a given period or after a certain number of iterations without improvement.
- The intensification of the search in a given region of the search space is not always useful.
- The effectiveness of the intensification depends on the landscape structure of the target optimization problem.

[1] Novelty Memory

Long-Term Memory/Diversification (1/2)

As mentioned many times, S-Metaheuristics are more powerful search methods in terms of intensification.
Long-Term Memory has been introduced in Tabu Search to encourage the diversification of the search.

- **The role of the Long-Term Memory is to force the search in non-explored regions of the search space.**
- **The main representation used for the Long-Term Memory is the Frequency Memory.**
- **As in the Recency Memory, the components associated with a solution have first to be defined.**
- **The Frequency Memory will memorize for each component the number of times the component is present in all visited solutions.**

Long-Term Memory/Diversification (2/2)

The diversification process can be applied periodically or after a given number of iterations without improvement.
Three popular diversification strategies may be applied:

- Restart diversification: introduces in the current or best solution the least visited components. Then a new search is restarted from this new solution.
- Continuous diversification: introduces during a search a bias[2] to encourage diversification (e.g., Record the frequency occurrence[3] of solution components in the evaluation of current solutions, then penalize frequently applied moves or visited solutions.
- Strategic oscillation: allows to consider (and penalize) intermediate solutions that are infeasible. This strategy will guide the search toward infeasible solutions and then come back to a feasible solution.

As for the intensification, the diversification of the search is not always useful. It depends on the landscape structure of the target optimization problem.

[2] Inclination or Prejudice
[3] The fact or frequency of something happening

The End