Tabu Search

禁忌搜索算法

吴婷钰

Outline

- Introduction
- Basic Tabu Search Algorithm
- An example of Tabu search
- Additional elements

Introduction

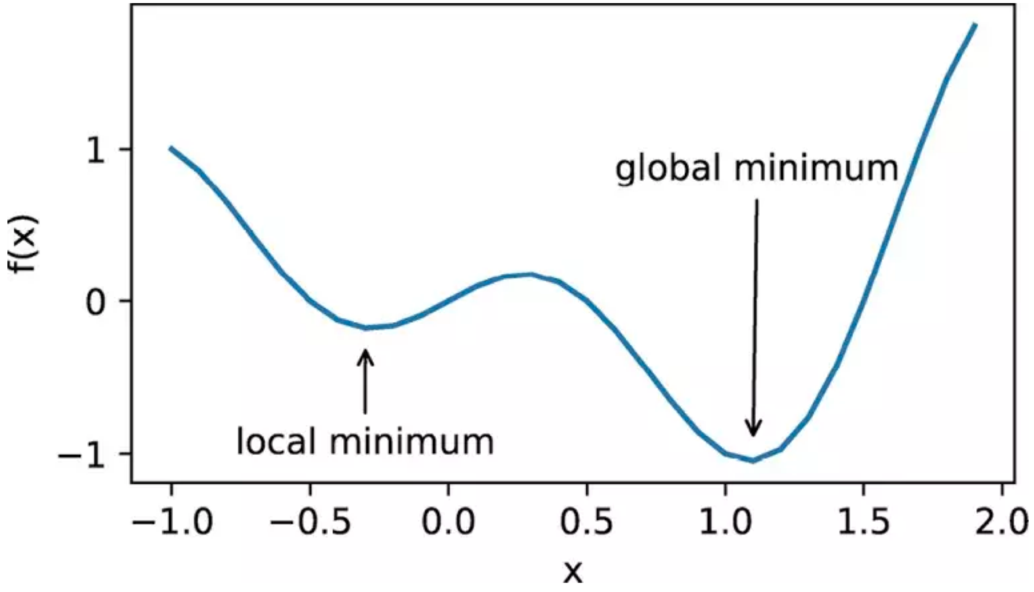
Introduction

- **Local Search**

- NP-hard Problem

- LS can be roughly summarized as an iterative search procedure:
  1. an initial feasible solution
  2. improve by applying local modifications
  3. terminate when reach a local optimum

- Limitations:
  1. sensitive to the initial solution
  2. depend on neighborhood structure
  3. no guarantee of global optimality



Introduction

- **Tabu Search** is based on introducing flexible memory structures in conjunction with strategic restrictions and aspiration levels as a mean for exploiting search spaces.

- It is a meta-heuristic that used to solve combinatorial optimization problems.

- The basic principle is to pursue LS whenever it encounters a local optimum by allowing non-improving moves; cycling back to previously visited solutions is prevented by the use of memories, called the Tabu List, that record the recent history of the search.

- It uses a flexible memory to restrict the next solution to some subsets of neighborhood of current solution.

- A dynamic neighborhood search method

Basic Tabu Search Algorithm

Tabu Search

- **3 main strategies:**
  - **Forbidding Strategy:** controls what enters the tabu list
  - **Freeing Strategy:** controls what exits from the tabu list
  - **Short-term Strategy:** manages interplay between the forbidding strategy and freeing strategy to select trial solutions

5	2	1	3	4

Distance = 50

5	2	4	3	1

Distance = 55 > 50 Tabu List = {(1,4)} (Tabu length = 3)

Basic Concepts of Tabu Search

- **Search Space:** the space of all possible solutions

- **Neighborhood construction:** to identify adjacent solutions that can be reached from current solution.

- A **tabu list** records forbidden/tabu moves which is a subset of the moves in the neighborhood construction.

- **Aspiration criterion:** allow to revoke (cancel) tabus, if it results in a solution with an objective value better than that of the current best-known solution. Tabus are sometimes too powerful, they may prohibit attractive moves, even when there is no danger of cycling, or they may lead to an overall stagnation of the searching process.

Basic Tabu Search Algorithm

- ***Initialization***

  Choose (construct) an initial solution S_0.

  Set $S \leftarrow S_0$, $f^* \leftarrow f(S_0)$, $S^* \leftarrow S_0$, $T \leftarrow \emptyset$.

- ***Search***

  While **termination criterion** not satisfied do

  Select S in $argmin_{S' \in \tilde{N}(S)}$ $[f(S')]$;

  if $f(S) < f^*$, then set $f^* \leftarrow f(S)$, $S^* \leftarrow S$;

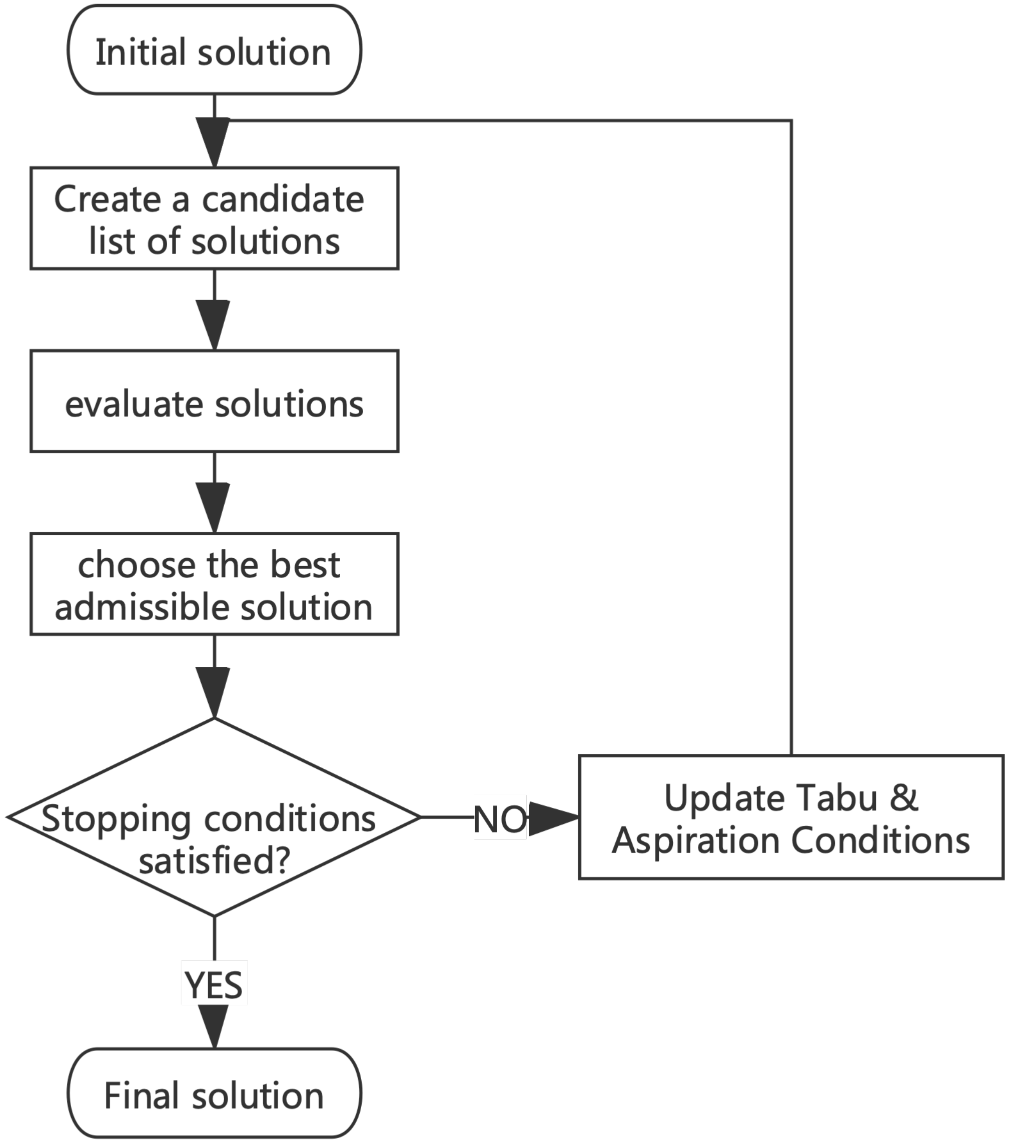
  record tabu for the current move in T (delete oldest entry if necessary).

  end While.

Terminal Criteria

- after a fixed number of iterations (or a fixed amount of CPU time);
- after some number of iterations without an improvement in the objective function value (**the criterion used in most implementations**);
- when the objective reaches a pre-specified threshold value.

Basic Tabu Search Algorithm Flow Chart



Advantages and Disadvantages

- **Advantages:**
  - Accepts non-improving solution to avoid local optimum.
  - Can be applied to both discrete ang continuous solution spaces.
  - Can be used for complex problems on scheduling.

- **Disadvantages:**
  - To many parameters to be determined.
  - Number of iterations could be too large.
  - Global optimum may not be found, depends on parameter settings.

An example of Tabu Search

Tabu Search

Example problem:

- 6 jobs are to be done in a sequence such that an Objective Function is maximized.

- Representation of a solution for 6 jobs

5	3	4	6	1	2

- Neighborhood structure: swapping of jobs

5	3	2	6	1	4

- A solution has 15 (6*5/2=15) neighbors (each of the 15 possible swapping of the 6 jobs)

Tabu Search

Recency Based Memory and Tabu Classification:

- 3 most recent jobs swaps are classified as tabu for the problem considered.
- Thus, the Tabu tenure will be for 3 iterations.
- Aspiration criterion is made use of to take some job swaps out of Tabu.

5	3	4	6	1	2

5	3	2	6	1	4

	2	3	4	5	6
1					
	2		3		
		3			
			4		
				5	

- With a (4,2) swap, we have (2,4) as Tabu for 3 iterations.

Tabu Search

Current Solution: Iteration 0

Current Solution

5	3	4	6	1	2

Obj Fn Value = 20

Tabu Structure

	2	3	4	5	6
1					
2					
3					
4					
5					

No entries

Top 4 Candidates

(5,4)	5 ★
(3,6)	3
(3,2)	1
(4,1)	-4

Tabu Search

Current Solution: Iteration 1

Current Solution

4	3	5	6	1	2

Obj Fn Value = 25

Tabu Structure

	2	3	4	5	6
1					
2					
3					
4				3	
5					

(4,5) for 3 iterations

Top 4 Candidates

(3,1)	3
(3,2)	1
(5,1)	-1
(6,2)	-5

Tabu Search

Current Solution: Iteration 2

Current Solution

4	1	5	6	3	2

Obj Fn Value = 28

Tabu Structure

	2	3	4	5	6
1		3			
2					
3					
4				2	
5					

(1,3) for 3 iterations
(4,5) for 2 iterations

Top 4 Candidates

(1,3)	-3
(4,2)	-4
(4,5)	-6
(6,2)	-7

Tabu Search

Current Solution: Iteration 3

Move (1,3) improves the Obj Fn. By 5

24 + 5 = 29 > 28

Hence, it is taken out of Tabu by **Aspiration Criteria**.

Current Solution

2	1	5	6	3	4

Obj Fn Value = 24

Tabu Structure

	2	3	4	5	6
1		2			
2					
3					
4				1	
5					

(1,3) for 2 iterations

(4,5) for 1 iterations

Top 4 Candidates

(1,3)	5
(2,5)	1
(1,4)	-1
(6,4)	-3

Tabu Search

Current Solution: Iteration 4

Current Solution

2	3	5	6	1	4

Obj Fn Value = 29

Tabu Structure

	2	3	4	5	6
1		3			
2					
3					
4					
5					

(3,1) for 3 iterations

Top 4 Candidates

(5,1)	0
(3,6)	-1
(1,4)	-2
(3,1)	-5

Move (5,1) does not improve Obj Fn.

Hence, the best solution obtained after 4 iterations is the current solution with Obj Fn = 29.

Additional elements

Additional elements

- Intensification
  1. Use intermediate-term memory (recency memory)
  2. Change the neighborhood structure to allowing more diverse moves

- Diversification

  Force the search into previously unexplored areas: use long-term memory (frequency memory):
  1. Restart diversification
  2. Continuous diversification
  3. Strategic oscillation

Additional elements

- Allowing Infeasible solutions
  1. Constraint relaxation: drop selected constrains and add weighted penalties (self-adjusting penalties: weights are adjusted dynamically on the basis of the recent history of the search)
  2. Strategic oscillation: modify penalty weights systematically to the search to cross the feasibility boundary of the search space and thus induce diversification.

- Surrogate and auxiliary objectives
  1. Evaluate neighbors using a surrogate objective
  2. Define an auxiliary objection function to orient the search

Thank You !