IE552 HEURISTIC METHODS FOR OPTIMIZATION

Lecture Notes 4 Tabu Search

Tabu Search

- Tabu search (TS) algorithm was proposed by Glover.
- In 1986, he pointed out the controlled randomization in SA to escape from local optima and proposed a deterministic algorithm.
- In a parallel work, a similar approach named “steepest ascent/mildest descent” has been proposed by Hansen.
- In the 1990s, the tabu search 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 tabu search.

Recall the classification

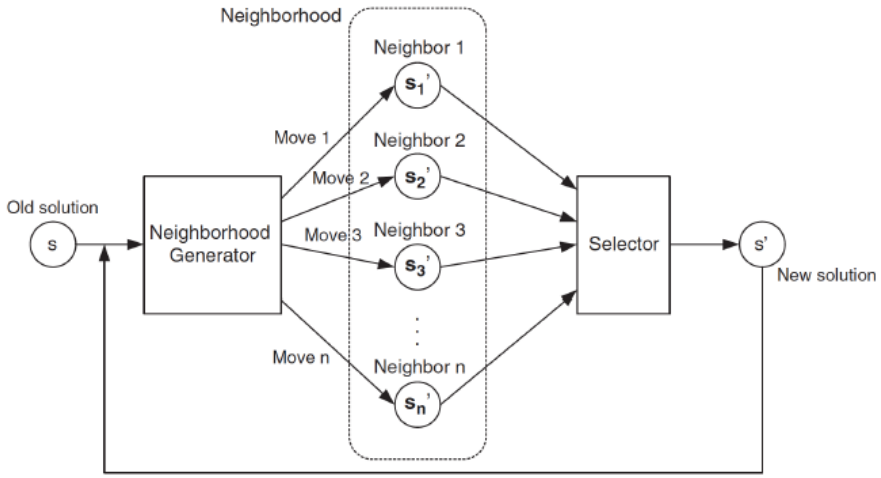
- Many classification criteria may be used for metaheuristics. The most common are:
  - Nature inspired vs non-nature inspired
  - Memory usage vs memoryless methods
  - Deterministic vs stochastic
  - Iterative vs greedy
  - Population-based search vs single-solution based search
- TS is single-solution based, deterministic, iterative, non-nature inspired and uses memory.

Tabu Search

- TS 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.
- What is steepest Local Search?
  - Start with a solution and go to the next one that gives the best improvement

Algorithm 6.1: Best Improvement Local Search

```
Create initial solution s;
while local optimum not reached do
    Determine complete neighborhood 𝒩 of current solution s;
    if 𝒩 contains at least one improving solution then
        Choose best solution s′ from 𝒩;
        Switch over to solution s′ (current solution s is replaced by s′);
    end
end
return s';
```

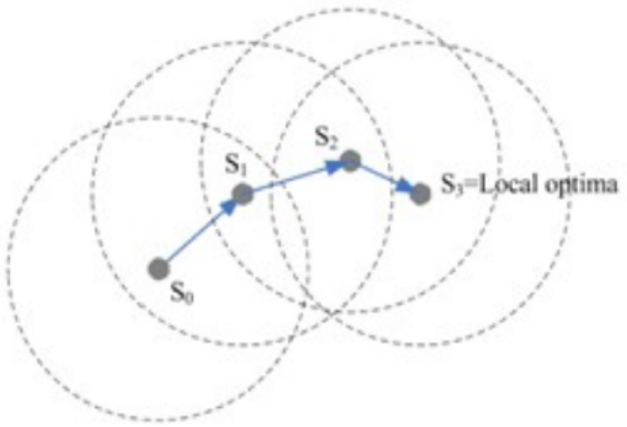


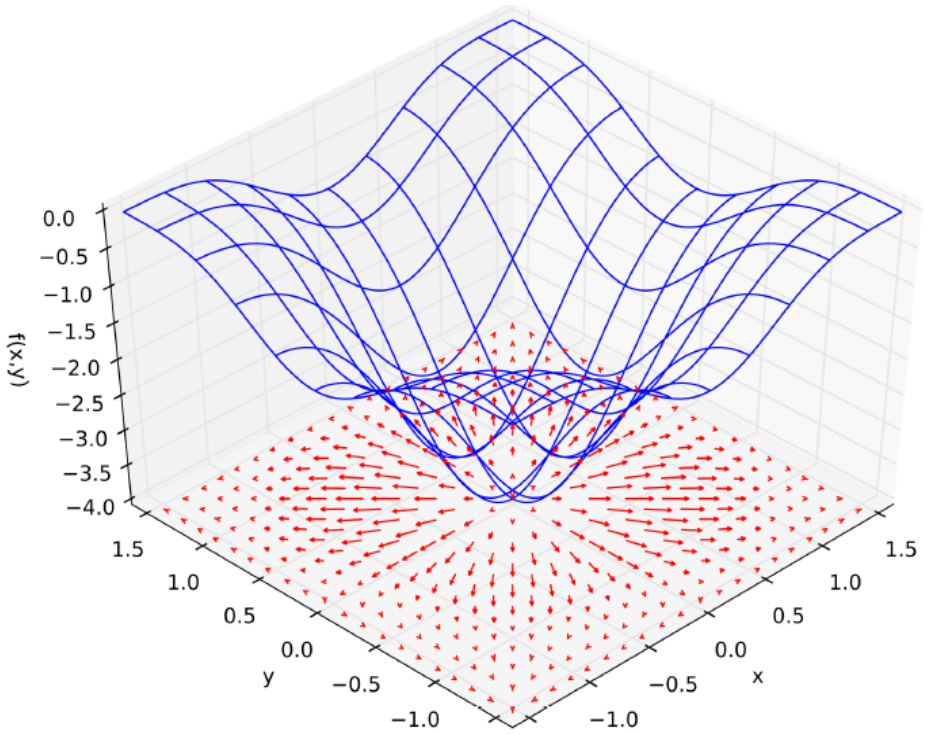
Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.
Zäpfel, G., & Braune, R. (2010). Metaheuristic search concepts: A tutorial with applications to production and logistics. Springer Science & Business Media.

Steepest Local Search (Steepest decent)

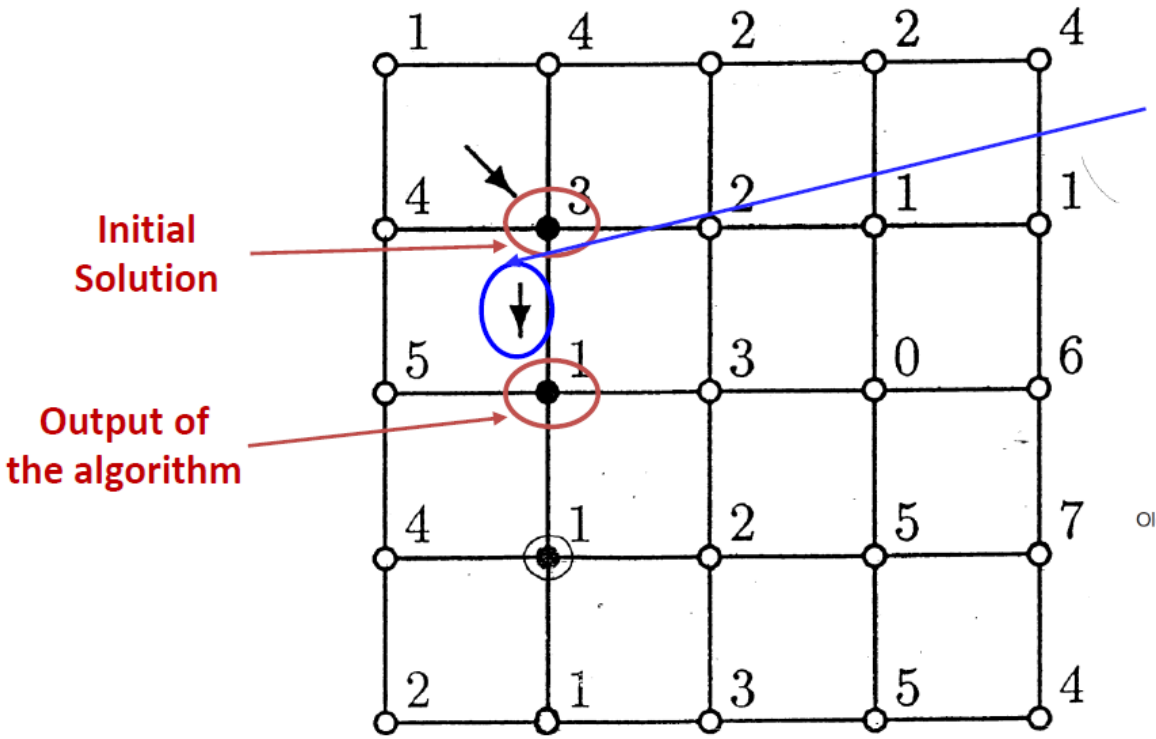
Gradient descent

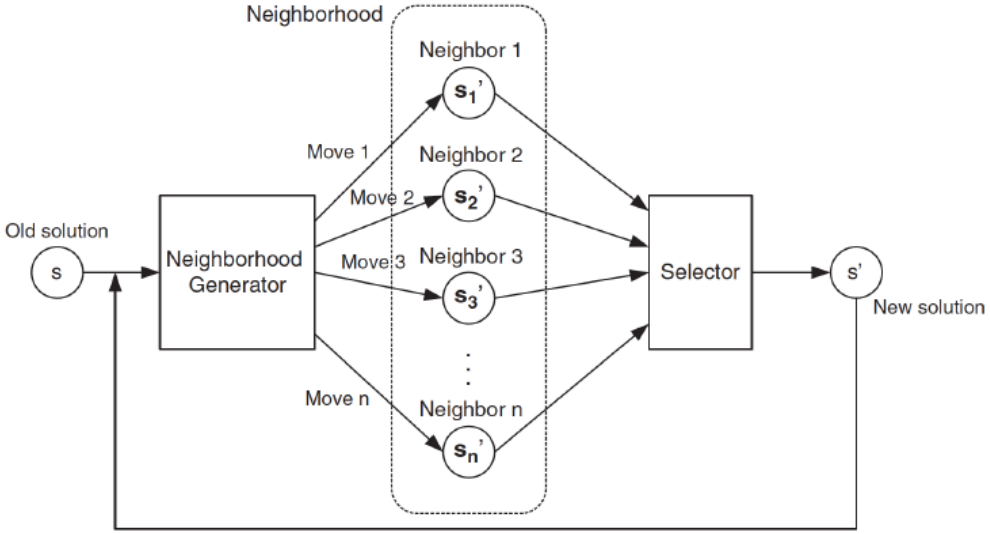
$x = x - \alpha * \nabla f(x)$





Steepest Local Search (Steepest decent)





Source: Exampled modified from http://sci2s.ugr.es/docencia/metah/

Tabu Search

- TS 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.
- Usually, the whole neighborhood is explored in a deterministic manner, whereas in SA a random neighbor is selected.
- As in local search, when a better neighbor is found, it replaces the current solution.
- When a local optima is reached, the search 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.
- Tabu search may be viewed as a dynamic transformation of the neighborhood.

Tabu Search

- This policy may generate cycles; that is, previous visited solutions could be selected again.
- To avoid cycles, TS discards the neighbors that have been previously visited.
- 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.
- The tabu list usually contains a constant number of tabu moves. Usually, the attributes of the moves are stored in the tabu list.

Tabu Search

- 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:** 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:** Aspiration criterion is a condition that can override the tabu status of a certain move. A commonly used aspiration criteria consists in selecting a tabu move if it generates a solution that is better than the best found solution.

Tabu Search Algorithm

Algorithm 2.9 Template of tabu search algorithm.

$s = s_0$; /* Initial solution */
Initialize the tabu list, medium-term and long-term memories ;
Repeat
  Find best admissible neighbor s' ; /* non tabu or aspiration criterion holds */
  $s = s'$;
  Update tabu list, aspiration conditions, medium and long term memories ;
  If intensification_criterion holds **Then** intensification ;
  If diversification_criterion holds **Then** diversification ;
Until Stopping criteria satisfied
Output: Best solution found.

Types of memory

The memory structures used in tabu search can roughly be divided into three categories:

- Short-term: The list of solutions recently considered. If a potential solution appears on the tabu list, it cannot be revisited until it reaches an expiration point.
- Intermediate-term: The medium-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. The search is biased by these attributes.
- Long-term: Diversification rules that drive the search into new regions (i.e. regarding resets when the search becomes stuck in a plateau or a suboptimal dead-end).

Types of memory

- The intensification and diversification processes can be applied periodically or after a given number of iterations without improvement.

Algorithm 2.9 Template of tabu search algorithm.

$s = s_0$; /* Initial solution */
Initialize the tabu list, medium-term and long-term memories ;
Repeat
  Find best admissible neighbor s' ; /* non tabu or aspiration criterion holds */
  $s = s'$;
  Update tabu list, aspiration conditions, medium and long term memories ;
  If intensification_criterion holds **Then** intensification ;
  If diversification_criterion holds **Then** diversification ;
Until Stopping criteria satisfied
Output: Best solution found.

Example 2.32 Tabu search for the TSP problem. Designing a TS for the TSP problem needs the definition of the tabu list (short-term memory), the medium- and long-term memories, and the aspiration criterion:

- The short-term memory maintains a list of t edges and prevents them from being selected for consideration of moves for a number of iterations l. After the given number of iterations (tabu tenure), these edges are released. One can use a 2D Boolean matrix to decide if an edge is tabu or not.

- The medium- and long-term memory maintains a list of t edges that have been considered in the last k best (worst) solutions. The process encourages (or discourages) their selection in future solutions using their frequency of appearance in the set of elite solutions and the quality of solutions they have appeared.
- The usual aspiration criterion that accepts the tabu moves that generate better solutions than the best found one.

Local (Improving) Search

1. Initialize: Choose a starting feasible solution, x^0 and t=0
2. If no move $\Delta x \in M$ is both improving and feasible, stop. x^t is local optimal. Otherwise choose some improving feasible move $\Delta x \in M$ update $\mathrm{x}^{\mathrm{t+1}} \leftarrow \ \Delta x + x^t \ and \ t \leftarrow t + 1$ Go to step 2.

Ex: $\max 20x_1 - 4x_2 + 34x_3$

$$st. 2x_1 + x_2 + 4x_3 \leq 5$$

$$x_1, x_2, x_3 \in \{0,1\}$$

The move set is $M = \{\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}-1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\-1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}, \begin{pmatrix}0\\0\\-1\end{pmatrix}\}$

Local (Improving) Search

Let starting feasible solution, $x^0 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$ and obj function value is 16.

There is only one move which is feasible and improving

$x^1 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ and obj function value is 20.

There is no feasible and improving move from here. So we are stuck at local optima. (optimal is clearly $x_3 = 1$ with objective 34)

- What we can do to avoid this?
  - We can do multistart: start with different feasible solutions & apply local search for all.
  - Or we can allow nonimproving moves!

Tabu Search: Small example

Ex: $\max 18x_1 + 25x_2 + 11x_3 + 14x_4$

$$st.\ 2x_1 + 2x_2 + x_3 + x_4 \leq 3$$

$$x_1, x_2, x_3 \in \{0,1\}$$

- Move: switch one 0-component to 1 or one 1-component to 0
- Tabu rule: if a component is switched, make it tabu for the next 2 iterations
- $t_{\max} = 5$
- $x^0 = (1,0,0,0)$

Tabu Search: Small example

Ex: $\max 18x_1 + 25x_2 + 11x_3 + 14x_4$

$$st.\ 2x_1 + 2x_2 + x_3 + x_4 \leq 3$$

$$x_1, x_2, x_3 \in \{0,1\}$$

- Move: switch one 0-component to 1 or one 1-component to 0
- Tabu rule: if a component is switched, make it tabu for the next 2 iterations
- $t_{\max} = 5$
- $x^0 = (1,0,0,0)$

t	x^t	Value	x^*	Tabu list
0	(1,0,0,0)	18	(1,0,0,0) 18	∅
1	(1,0,0,1)	32	(1,0,0,1) 32	{4}
2	(0,0,0,1)	14	(1,0,0,1) 32	{4,1}
3	(0,1,0,1)	39	(0,1,0,1) 39	{1,2}
4	(0,1,0,0)	25	(0,1,0,1) 39	{2,4}
5	(0,1,1,0)	36	(0,1,0,1) 39	{4,3}