METAHEURISTICS
Neighborhood (Local) Search Techniques

Jacques A. Ferland
Department of Informatique and Recherche Opérationnelle
Université de Montréal

ferland@iro.umontreal.ca
Introduction

- Introduction to some basic methods
- My vision relying on my experience
- Not an exhaustive survey
- Most of the time ad hoc adaptations of basic methods are used to deal with specific applications
Advantages of using metaheuristic

• Intuitive and easy to understand

• With regards to enduser of a real world application:
  - Easy to explain
  - Connection with the manual approach of enduser
  - Enduser sees easily the added features to improve the results
  - Allow to analyze more deeply and more scenarios

• Allow dealing with larger size problems having higher degree of complexity

• Generate rapidly very good solutions
Disadvantages of using metaheuristic

- Quick and dirty methods
- Optimality not guaranteed in general
- Few convergence results for special cases
Summary

• Heuristic Constructive techniques:
  Greedy
  GRASP (Greedy Randomized Adaptive Search Procedure)

• Neighborhood (Local) Search Techniques:
  Descent
  Tabu Search
  Simulated Annealing
  Threshold Accepting

• Improving strategies
  Intensification
  Diversification
  Variable Neighborhood Search (VNS)
  Exchange Procedure
Problem used to illustrate

- **General problem**

  \[
  \min f(x) \\
  x \in X
  \]

- **Assignment type problem:** Assignment of resources \( j \) to activities \( i \)

  \[
  \min f(x) \\
  \text{Subject to } \sum_{1 \leq j \leq m} x_{ij} = 1 \quad 1 \leq i \leq n \\
  x_{ij} = 0 \text{ or } 1 \quad 1 \leq i \leq n, \ 1 \leq j \leq m
  \]
Problem Formulation

- **Assignment type problem**

  \[
  \min f(x) \\
  \text{Subject to } \sum_{1 \leq j \leq m} x_{ij} = 1 \quad 1 \leq i \leq n \\
  x_{ij} = 0 \text{ or } 1 \quad 1 \leq i \leq n, \ 1 \leq j \leq m
  \]

- **Graph coloring problem** : Graph \( G = (V,E) \).

  \( V = \{ i : 1 \leq i \leq n \} \) ; \( E = \{(i, l) : (i, l) \text{ edge of } G\} \).
  
  Set of colors \( \{ j : 1 \leq j \leq m \} \)

  \[
  \min \sum_{1 \leq j \leq m} \sum_{(i, l) \in E} x_{ij} x_{lj} \\
  \text{Subject to } \sum_{1 \leq j \leq m} x_{ij} = 1 \quad 1 \leq i \leq n \\
  x_{ij} = 0 \text{ or } 1 \quad 1 \leq i \leq n, \ 1 \leq j \leq m
  \]
Graph coloring example

- **Graph coloring problem**: Graph $G = (V,E)$.
  
  $V = \{ i : 1 \leq i \leq n \}$ ; $E = \{(i, l) : (i, l) \text{ edge of } G\}$.
  
  Set of colors $\{j : 1 \leq j \leq m\}$

  $\text{min } f(x) = \sum_{1 \leq j \leq m} \sum_{(i, l) \in E} x_{ij} x_{lj}$

  Subject to $\sum_{1 \leq j \leq m} x_{ij} = 1 \quad 1 \leq i \leq n$

  $x_{ij} = 0 \text{ or } 1 \quad 1 \leq i \leq n, \ 1 \leq j \leq m$

- To simplify notation, denote or encode a solution $x$ as follows:

  $x \Rightarrow [j(1), j(2), ..., j(i), ..., j(n)]$

  where for each vertex $i$,

  $x_{ij(i)} = 1$

  $x_{ij} = 0 \quad \text{for all other } j$
Heuristic Constructive Techniques

• Values of the variables are determined sequentially: at each iteration, a variable is selected, and its value is determined

• The value of each variable is never modified once it is determined

• Techniques often used to generate initial solutions for iterative procedures
Greedy method

Next variable to be fixed and its value are selected to optimize the objective function given the values of the variables already fixed

Graph coloring problem:

Vertices are ordered in decreasing order of their degree

Vertices selected in that order

For each vertex, select a color in order to reduce the number of pairs of adjacent vertices already colored with the same color
Graph coloring example

- Graph with 5 vertices
- 2 colors available: red blue
Graph coloring example

- Vertices in decreasing order of degree
- Vertex | degree | color
  2     | 3      |
  1     | 2      |
  3     | 2      |
  4     | 2      |
  5     | 1      |
Graph coloring example

- Vertices in decreasing order of degree
- Vertex | degree | color
  2      | 3      | red
  1      | 2      | blue
  3      | 2
  4      | 2
  5      | 1

- Vertex 2

<table>
<thead>
<tr>
<th>color</th>
<th>number of adj. vert. same color</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>0</td>
</tr>
<tr>
<td>blue</td>
<td>0 &lt;=</td>
</tr>
</tbody>
</table>
Graph coloring example

- Vertices in decreasing order of degree
- Vertex | degree | color
  - 2     | 3      | blue
  - 1     | 2      |
  - 3     | 2      |
  - 4     | 2      |
  - 5     | 1      |

- Vertex 2

<table>
<thead>
<tr>
<th>color</th>
<th>number of adj. vert. same color</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>0</td>
</tr>
<tr>
<td>blue</td>
<td>0 &lt;=</td>
</tr>
</tbody>
</table>
Graph coloring example

- Vertices in decreasing order of degree
- Vertex | degree | color
  2      | 3      | blue
  1      | 2      |
  3      | 2      |
  4      | 2      |
  5      | 1      |
Graph coloring example

- Vertices in decreasing order of degree
- Vertex | degree | color
  |       |       |
  2     | 3     | blue
  1     | 2     |
  3     | 2     |
  4     | 2     |
  5     | 1     |

- Vertex 1

<table>
<thead>
<tr>
<th>color</th>
<th>number of adj. vert. same color</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>0</td>
</tr>
<tr>
<td>blue</td>
<td>1</td>
</tr>
</tbody>
</table>
Graph coloring example

- Vertices in decreasing order of degree
- Vertex | degree | color
  2     | 3      | blue
  1     | 2      | red
  3     | 2      |
  4     | 2      |
  5     | 1      |

- Vertex 1

<table>
<thead>
<tr>
<th>color</th>
<th>number of adj. vert. same color</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>0</td>
</tr>
<tr>
<td>blue</td>
<td>1</td>
</tr>
</tbody>
</table>
Graph coloring example

- Vertices in decreasing order of degree
- | Vertex | degree | color |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>blue</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>red</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Graph coloring example

- Vertices in decreasing order of degree

<table>
<thead>
<tr>
<th>Vertex</th>
<th>degree</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>blue</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>red</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

- Vertex 3

<table>
<thead>
<tr>
<th>color</th>
<th>number of adj. vert. same color</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>1 &lt;=</td>
</tr>
<tr>
<td>blue</td>
<td>1</td>
</tr>
</tbody>
</table>
Graph coloring example

- Vertices in decreasing order of degree
- Vertex | degree | color
  2 3 | blue
  1 2 | red
  3 2 | red
  4 2 |
  5 1 |

- Vertex 3

<table>
<thead>
<tr>
<th>color</th>
<th>number of adj. vert. same color</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>1 &lt;=</td>
</tr>
<tr>
<td>blue</td>
<td>1</td>
</tr>
</tbody>
</table>
Graph coloring example

- Vertices in decreasing order of degree
- Vertex | degree | color
  2     | 3     | blue
  1     | 2     | red
  3     | 2     | red
  4     | 2     |
  5     | 1     |
Graph coloring example

- Vertices in decreasing order of degree
- Vertex | degree | color
          |       |    
2        | 3     | blue
1        | 2     | red
3        | 2     | red
4        | 2     | 
5        | 1     | 

- Vertex 4

<table>
<thead>
<tr>
<th>color</th>
<th>number of adj. vert. same color</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>0 &lt;=</td>
</tr>
<tr>
<td>blue</td>
<td>1</td>
</tr>
</tbody>
</table>
Graph coloring example

- Vertices in decreasing order of degree

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Degree</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>blue</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>red</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>red</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>red</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

- Vertex 4

<table>
<thead>
<tr>
<th>Color</th>
<th>Number of Adj. Vert. Same Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>0 &lt;=</td>
</tr>
<tr>
<td>blue</td>
<td>1</td>
</tr>
</tbody>
</table>
Graph coloring example

- Vertices in decreasing order of degree
- Vertex | degree | color
- 2     | 3      | blue
- 1     | 2      | red
- 3     | 2      | red
- 4     | 2      | red
- 5     | 1      |
Graph coloring example

- Vertices in decreasing order of degree
- Vertex | degree | color
  | 2 | 3 | blue
  | 1 | 2 | red
  | 3 | 2 | red
  | 4 | 2 | red
  | 5 | 1 |

- Vertex 5

<table>
<thead>
<tr>
<th>color</th>
<th>number of adj. vert. same color</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>1</td>
</tr>
<tr>
<td>blue</td>
<td>0 &lt;=</td>
</tr>
</tbody>
</table>
Graph coloring example

- Vertices in decreasing order of degree
- Vertex | degree | color
  --- | --- | ---
  2 | 3  | blue
  1 | 2  | red
  3 | 2  | red
  4 | 2  | red
  5 | 1  | blue

- Vertex 5

<table>
<thead>
<tr>
<th>color</th>
<th>number of adj. vert. same color</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>1</td>
</tr>
<tr>
<td>blue</td>
<td>0 &lt;=</td>
</tr>
</tbody>
</table>
GRASP
Greedy Randomized Adaptive Search Procedure

- Next variable to be fixed is selected randomly among those inducing the smallest increase.
- Referring to the general problem,
  i) let \( J' = \{ j : x_j \text{ is not fixed yet} \} \)
    and \( \delta_j \) be the increase induces by the best value that \( x_j \) can take (\( j \in J' \))

  ii) Denote \( \delta^* = \min_{j \in J'} \{ \delta_j \} \)
      and \( \alpha \in [0, 1] \).

  iii) Select randomly \( j' \in \{ j \in J' : \delta_j \leq (1/\alpha) \delta^* \} \)
      and fix the value of \( x_{j'} \).
Graph coloring example

- Graph with 5 vertices
- 2 colors available: red blue
- \( \alpha = 0.5 \)
Graph coloring example

<table>
<thead>
<tr>
<th>Vertex $j$</th>
<th>best color</th>
<th>$\delta_j$</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Graph coloring example

<table>
<thead>
<tr>
<th>Vertex $j$</th>
<th>best color</th>
<th>$\delta_j$</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

$J' = \{1, 2, 3, 4, 5\}$

$\delta^* = 0$

$\alpha = 0.5$

$\{j \in J' : \delta_j \leq (1 / \alpha) \delta^* \} = \{1, 2, 3, 4, 5\}$

Select randomly vertex 3

color blue
Graph coloring example

<table>
<thead>
<tr>
<th>Vertex $j$</th>
<th>best color</th>
<th>$\delta_j$</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>any</td>
<td>0</td>
<td>blue</td>
</tr>
<tr>
<td>4</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

$J' = \{1, 2, 3, 4, 5\}$

$\delta^* = 0$

$\alpha = 0.5$

$\{j \in J' : \delta_j \leq (1 / \alpha) \delta^* \} = \{1, 2, 3, 4, 5\}$

- Select randomly vertex 3
  - color blue
Graph coloring example

<table>
<thead>
<tr>
<th>Vertex $j$</th>
<th>best color</th>
<th>$\delta_j$</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>red</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>red</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>any</td>
<td></td>
<td>blue</td>
</tr>
<tr>
<td>4</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Graph coloring example

<table>
<thead>
<tr>
<th>Vertex $j$</th>
<th>best color</th>
<th>$\delta_j$</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>red</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>red</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>blue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

$J' = \{1, 2, 4, 5\}$

$\delta^* = 0$

$\alpha = 0.5$

$\{j \in J': \delta_j \leq (1/\alpha) \delta^* \} = \{1, 2, 4, 5\}$

- Select randomly vertex 4
  - color red
Graph coloring example

<table>
<thead>
<tr>
<th>Vertex $j$</th>
<th>best color</th>
<th>$\delta_j$</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>red</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>red</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>red</td>
<td>0</td>
<td>blue</td>
</tr>
<tr>
<td>4</td>
<td>any</td>
<td>0</td>
<td>red</td>
</tr>
<tr>
<td>5</td>
<td>any</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

$J' = \{1, 2, 4, 5\}$

$\delta^* = 0$

$\alpha = 0.5$

$\{j \in J' : \delta_j \leq \left(\frac{1}{\alpha}\right) \delta^*\} = \{1, 2, 4, 5\}$

- Select randomly vertex 4
  - color red
Graph coloring example

<table>
<thead>
<tr>
<th>Vertex $j$</th>
<th>best color</th>
<th>$\delta_j$</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>red</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>any</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>blue</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>red</td>
</tr>
<tr>
<td>5</td>
<td>blue</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Graph coloring example

<table>
<thead>
<tr>
<th>Vertex $j$</th>
<th>best color</th>
<th>$\delta_j$</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>red</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>any</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>blue</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>blue</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

$J' = \{1, 2, 5\}$

$\delta^* = 0$

$\alpha = 0.5$

$\{j \in J' : \delta_j \leq (1 / \alpha) \delta^* \} = \{1, 5\}$

➢ Select randomly vertex 5
color blue
Graph coloring example

<table>
<thead>
<tr>
<th>Vertex $j$</th>
<th>best color</th>
<th>$\delta_j$</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>red</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>any</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>blue</td>
</tr>
<tr>
<td>4</td>
<td>red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>blue</td>
<td>0</td>
<td>blue</td>
</tr>
</tbody>
</table>

$J' = \{1, 2, 5\}$
$\delta^* = 0$
$\alpha = 0.5$

$\{j \in J' : \delta_j \leq (1 / \alpha) \delta^* \} = \{1, 5\}$

- Select randomly vertex 5 color blue
Graph coloring example

<table>
<thead>
<tr>
<th>Vertex $j$</th>
<th>best color</th>
<th>$\delta_j$</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>red</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>any</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>blue</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>red</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>blue</td>
</tr>
</tbody>
</table>
Graph coloring example

<table>
<thead>
<tr>
<th>Vertex $j$</th>
<th>best color</th>
<th>$\delta_j$</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>red</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>any</td>
<td>1</td>
<td>blue</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>red</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>blue</td>
</tr>
</tbody>
</table>

$J' = \{1, 2\}$

$\delta^* = 0$

$\alpha = 0.5$

$\{j \in J' : \delta_j \leq (1/\alpha) \delta^* \} = \{1\}$

➢ Select vertex 1

color red
Graph coloring example

<table>
<thead>
<tr>
<th>Vertex $j$</th>
<th>best color</th>
<th>$\delta_j$</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>red</td>
<td>0</td>
<td>red</td>
</tr>
<tr>
<td>2</td>
<td>any</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>blue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>blue</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$J' = \{1, 2\}$

$\delta^* = 0$

$\alpha = 0.5$

$\{j \in J' : \delta_j \leq (1 / \alpha) \delta^* \} = \{1\}$

➢ Select vertex 1

color red
Graph coloring example

<table>
<thead>
<tr>
<th>Vertex $j$</th>
<th>best color</th>
<th>$\delta_j$</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>red</td>
</tr>
<tr>
<td>2</td>
<td>blue</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>blue</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>red</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>blue</td>
</tr>
</tbody>
</table>
Graph coloring example

Vertex $j$ | best color | $\delta_j$ | color
---|---|---|---
1 | | | red
2 | blue | 1 | blue
3 | | | blue
4 | | | red
5 | | | blue

$J' = \{2\}$

$\delta^* = 1$

$\alpha = 0.5$

$\{j \in J' : \delta_j \leq (1/\alpha) \delta^* \} = \{2\}$

- Select vertex 2
  - color blue
Graph coloring example

<table>
<thead>
<tr>
<th>Vertex $j$</th>
<th>best color</th>
<th>$\delta_j$</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>blue</td>
<td>1</td>
<td>blue</td>
</tr>
<tr>
<td>3</td>
<td>blue</td>
<td></td>
<td>blue</td>
</tr>
<tr>
<td>4</td>
<td>red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>blue</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$J' = \{2\}$  
$\delta^* = 1$  
$\alpha = 0.5$  
$\{j \in J' : \delta_j \leq \left( \frac{1}{\alpha} \right) \delta^* \} = \{2\}$  

➢ Select vertex 2  
    color blue
Graph coloring example

<table>
<thead>
<tr>
<th>Vertex $j$</th>
<th>best color</th>
<th>$\delta_j$</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>red</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>blue</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>blue</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>red</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>blue</td>
</tr>
</tbody>
</table>

$J' = \Phi$
Neighborhood (Local) Search Techniques (NST)

• A Neighborhood (Local) Search Technique (NST) is an iterative procedure starting with an initial feasible solution $x^0$.

• At each iteration:
  - we move from the current solution $x \in X$ to a new one $x' \in X$ in its neighborhood $N(x)$
  - $x'$ becomes the current solution for the next iteration
  - we update the best solution $x^*$ found so far.

• The procedure continues until some stopping criterion is satisfied
Neighborhood

Neighborhood $N(x)$:

The neighborhood $N(x)$ varies with the problem, but its elements are always generated by slightly modifying $x$.

If we denote $M$ the set of modifications (or moves) to generate neighboring solutions, then

$$N(x) = \{ x' : x' = x \oplus mo , \; mo \in M \}$$
Neighborhood for assignment type problem

• For the assignment type problem:
  Let $x$ be as follows: for each $1 \leq i \leq n$,
  
  $x_{ij(i)} = 1$
  $x_{ij} = 0$ for all other $j$

  Each solution $x' \in N(x)$ is obtained by selecting an activity $i$ and modifying its resource from $j(i)$ to some other $p$
  (i. e., the modification can be denoted $mo = [i, p]$):
  
  $x'_{ij(i)} = 0$
  $x'_{ip} = 1$
  $x'_{ij} = x_{ij}$ for all other $i, j$

The elements of the neighborhood $N(x)$ are generated by slightly modifying $x$:

$N(x) = \{ x' : x' = x \oplus mo , mo \in M \}$
Neighborhood (Local) Search Techniques (NST)

- Descent method
- Tabu Search
- Simulated Annealing
- Threshold Accepting

- Introduce these methods using pseudo-codes
Descent Method ($D$)

• At each iteration, a best solution $x' \in N(x)$ is selected as the current solution for the next iteration.

• Stopping criterion:

$$f(x') \geq f(x)$$

i.e., the current solution cannot be improved or a first local minimum is reached.
Selecting $x'$
Selecting $x'$
Selecting $x'$
Selecting $x'$
Descent Method \((D)\)

- **Initialize**
  Select an initial solution \(x^0 \in X\)
  Let \(x := x^0\)
Descent Method ($D$)

- **Initialize**
  Select an initial solution $x^0 \in X$
  Let $x := x^0$ and $stop := false$

- **While not stop**
  Determine $x' \in N(x)$ such that
  
  $$x' := \text{argmin}_{z \in N(x)} \{ f(z) \}$$

At each iteration, a best solution $x' \in N(x)$ is selected
Descent Method ($D$)

- **Initialize**
  Select an initial solution $x^0 \in X$
  Let $x := x^0$ and $stop := false$

- **While** not $stop$
  Determine $x' \in N(x)$ such that
  \[
  x' := \arg\min_{z \in N(x)} \{ f(z) \}
  \]
  \[
  \text{If } f(x') \geq f(x) \quad \text{then } stop := true
  \]

- $x$ is the solution generated

At each iteration, a best solution $x' \in N(x)$ is selected

Stopping criterion

a first local minimum is reached
Descent Method ($D$)

- **Initialize**
  Select an initial solution $x^0 \in X$
  Let $x := x^0$ and $stop := false$

- **While** not $stop$
  Determine $x' \in N(x)$ such that
  
  $$x' := \operatorname{argmin}_{z \in N(x)} \{ f(z) \}$$
  
  If $f(x') \geq f(x)$
  then $stop := true$
  else $x := x'$

- $x$ is the solution generated

At each iteration, a best solution $x' \in N(x)$ is selected

$x' \in N(x)$ is the current solution for the next iteration
Graph coloring example

• Graph coloring problem: Graph $G = (V, E)$.
  $V = \{i : 1 \leq i \leq n\}$; $E = \{(i, l) : (i, l) \text{ edge of } G\}$.
  Set of colors $\{j : 1 \leq j \leq m\}$

$$
\begin{align*}
\min f(x) &= \sum_{1 \leq j \leq m} \sum_{(i, l) \in E} x_{ij} x_{lj} \\
\text{Subject to} & \sum_{1 \leq j \leq m} x_{ij} = 1 & 1 \leq i \leq n \\
& x_{ij} = 0 \text{ or } 1 & 1 \leq i \leq n, \ 1 \leq j \leq m
\end{align*}
$$

• To simplify notation, denote or encode a solution $x$ as follows:

$$
x = \Rightarrow [j(1), j(2), \ldots, j(i), \ldots, j(n)]
$$

where for each vertex $i$,

$$
\begin{align*}
x_{ij(i)} &= 1 \\
x_{ij} &= 0 \quad \text{for all other } j
\end{align*}
$$
Graph coloring example

Initial sol. $x^0 = x \Rightarrow [b, b, r, r, r] ; f(x) = 2$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
Graph coloring example

Initial sol. $x^0 = x \Rightarrow [b, b, r, r, r] ; f(x) = 2$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[r, b, r, r, r]</td>
<td>2</td>
</tr>
</tbody>
</table>
Graph coloring example

Initial sol. $x^0 = x \Rightarrow [b, b, r, r, r] ; f(x) = 2$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[r, b, r, r, r]</td>
<td>2</td>
</tr>
<tr>
<td>[b, r, r, r, r]</td>
<td>3</td>
</tr>
</tbody>
</table>
Graph coloring example

Initial sol. $x^0 = x \Rightarrow [b, b, r, r, r]$ ; $f(x) = 2$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[r, b, r, r, r]</td>
<td>2</td>
</tr>
<tr>
<td>[b, r, r, r, r]</td>
<td>3</td>
</tr>
<tr>
<td>[b, b, b, r, r]</td>
<td>4</td>
</tr>
</tbody>
</table>
Graph coloring example

Initial sol. \( x^0 = x \mapsto [b, b, r, r, r] ; f(x) = 2 \)

<table>
<thead>
<tr>
<th>( N(x) )</th>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( [r, b, r, r, r] )</td>
<td>2</td>
</tr>
<tr>
<td>( [b, r, r, r, r] )</td>
<td>3</td>
</tr>
<tr>
<td>( [b, b, b, r, r] )</td>
<td>4</td>
</tr>
<tr>
<td>( [b, b, b, b, r] )</td>
<td>2</td>
</tr>
</tbody>
</table>
Graph coloring example

Initial sol. $x^0 = x \Rightarrow [b, b, r, r, r] ; f(x) = 2$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[r, b, r, r]</td>
<td>2</td>
</tr>
<tr>
<td>[b, r, r, r]</td>
<td>3</td>
</tr>
<tr>
<td>[b, b, b, r]</td>
<td>4</td>
</tr>
<tr>
<td>[b, b, r, b]</td>
<td>2</td>
</tr>
<tr>
<td>[b, b, r, b]</td>
<td>1</td>
</tr>
</tbody>
</table>
Graph coloring example

Initial sol. $x^0 = x \Rightarrow [b, b, r, r, r] ; f(x) = 2$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[r, b, r, r, r]</td>
<td>2</td>
</tr>
<tr>
<td>[b, r, r, r, r]</td>
<td>3</td>
</tr>
<tr>
<td>[b, b, b, r, r]</td>
<td>4</td>
</tr>
<tr>
<td>[b, b, r, b, r]</td>
<td>2</td>
</tr>
<tr>
<td>[b, b, r, r, b]</td>
<td>1</td>
</tr>
</tbody>
</table>

- $x' \Rightarrow [b, b, r, r, b] ; f(x') = 1$
- Since $f(x') < f(x)$, then
  $x'$ replaces the current sol. $x$
  and we start a new iteration
Graph coloring example

Current sol. $x \Rightarrow [b, b, r, r, b]$; $f(x) = 1$
Graph coloring example

Current sol. $x \Rightarrow [b, b, r, r, b] ; f(x) = 1$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[r, b, r, r, b]$</td>
<td>1</td>
</tr>
</tbody>
</table>
Graph coloring example

Current sol. $x \Rightarrow [b, b, r, r, b] ; f(x) = 1$

\[
\begin{array}{|c|c|}
\hline
N(x) & f \\
\hline
[r, b, r, r, b] & 1 \\
[b, r, r, r, b] & 2 \\
\hline
\end{array}
\]
Graph coloring example

Current sol. $x \Rightarrow [b, b, r, r, b]; f(x) = 1$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[r, b, r, r, b]</td>
<td>1</td>
</tr>
<tr>
<td>[b, r, r, r, b]</td>
<td>2</td>
</tr>
<tr>
<td>[b, b, b, r, b]</td>
<td>3</td>
</tr>
</tbody>
</table>
Graph coloring example

Current sol. $x \Rightarrow [b, b, r, r, b] ; f(x) = 1$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[r, b, r, r, b]</td>
<td>1</td>
</tr>
<tr>
<td>[b, r, r, r, b]</td>
<td>2</td>
</tr>
<tr>
<td>[b, b, b, r, b]</td>
<td>3</td>
</tr>
<tr>
<td>[b, b, b, b, b]</td>
<td>3</td>
</tr>
</tbody>
</table>
Graph coloring example

Current sol. $x => [b, b, r, r, b]$ ; $f(x) = 1$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[r, b, r, r, b]</td>
<td>1</td>
</tr>
<tr>
<td>[b, r, r, r, b]</td>
<td>2</td>
</tr>
<tr>
<td>[b, b, b, r, b]</td>
<td>3</td>
</tr>
<tr>
<td>[b, b, r, b, b]</td>
<td>3</td>
</tr>
<tr>
<td>[b, b, r, b, b]</td>
<td>3</td>
</tr>
<tr>
<td>[b, b, r, r, r]</td>
<td>2</td>
</tr>
</tbody>
</table>
Graph coloring example

Current sol. \( x \Rightarrow [b, b, r, r, b] ; f(x) = 1 \)

<table>
<thead>
<tr>
<th>( N(x) )</th>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[r, b, r, r, b]</td>
<td>1</td>
</tr>
<tr>
<td>[b, r, r, r, b]</td>
<td>2</td>
</tr>
<tr>
<td>[b, b, b, r, b]</td>
<td>3</td>
</tr>
<tr>
<td>[b, b, r, b, b]</td>
<td>3</td>
</tr>
<tr>
<td>[b, b, r, r]</td>
<td>2</td>
</tr>
</tbody>
</table>

\( x' \Rightarrow [r, b, r, r, b] ; f(x') = 1 \)

- Since \( f(x') = f(x) \), then the procedure stops with \( x^* = x \Rightarrow [b, b, r, r, b] \)
Second example

Initial sol. $x^0 = x \Rightarrow [r, r, b, b] ; f(x) = 2$

\[
\begin{array}{c|c|c|c|c}
N(x) & f \\
\hline
\end{array}
\]
Second example

Initial sol. $x^0 = x \Rightarrow [r, r, b, b] ; f(x) = 2$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[b, r, b, b]$</td>
<td>2</td>
</tr>
</tbody>
</table>
Second example

Initial sol. \( x^0 = x \Rightarrow [r, r, b, b] ; f(x) = 2 \)

<table>
<thead>
<tr>
<th>( N(x) )</th>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[b, r, b, b]</td>
<td>2</td>
</tr>
<tr>
<td>[r, b, b, b]</td>
<td>2</td>
</tr>
</tbody>
</table>
Second example

Initial sol. $x^0 = x \Rightarrow [r, r, b, b] ; f(x) = 2$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[b, r, b, b]$</td>
<td>2</td>
</tr>
<tr>
<td>$[r, b, b, b]$</td>
<td>2</td>
</tr>
<tr>
<td>$[r, r, r, b]$</td>
<td>2</td>
</tr>
</tbody>
</table>
Second example

Initial sol. $x^0 = x \Rightarrow [r, r, b, b] ; f(x) = 2$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[b, r, b, b]</td>
<td>2</td>
</tr>
<tr>
<td>[r, b, b, b]</td>
<td>2</td>
</tr>
<tr>
<td>[r, r, r, b]</td>
<td>2</td>
</tr>
<tr>
<td>[r, r, b, r]</td>
<td>2</td>
</tr>
</tbody>
</table>
Second example

Initial sol. \( x^0 = x \Rightarrow [r, r, b, b] ; f(x) = 2 \)

<table>
<thead>
<tr>
<th>( N(x) )</th>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[b, r, b, b]</td>
<td>2</td>
</tr>
<tr>
<td>[r, b, b, b]</td>
<td>2</td>
</tr>
<tr>
<td>[r, r, r, b]</td>
<td>2</td>
</tr>
<tr>
<td>[r, r, b, r]</td>
<td>2</td>
</tr>
</tbody>
</table>

- \( x' \) is any neighboring solution
- \( f(x') = 2 \)
- Since \( f(x') = f(x) \), then the procedure stops with \( x^* = x \)
Second example

• But the solution
  \[ x \Rightarrow [r, b, r, b] \]
  is feasible and \( f(x) = 0 \).

• Hence the descent method is not able to reach the global minimum using the initial solution
  \[ [r, r, b, b] \]
  and the current definition of the neighborhood.
Knapsack Problem

• Problem formulation:

\[\max f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4\]
Subject to \[2x_1 + 2x_2 + x_3 + x_4 \leq 3 \quad (*)\]
\[x_1, x_2, x_3, x_4 = 0 \text{ or } 1.\]

• Neighborhood \(N(x)\) specified by the following modification or move:

The value of one and only one variable is modified (from 0 to 1 or from 1 to 0) to generate a new solution satisfying constraint
\[2x_1 + 2x_2 + x_3 + x_4 \leq 3 \quad (*)\]
Knapsack Problem

\[ \text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4 \]

Subject to

\[ 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \]

\[ x_1, x_2, x_3, x_4 = 0 \text{ or } 1. \]

• Initial sol. \( x^0 = x = [1, 0, 0, 0] \);
  \[ f(x) = 18 \]

<table>
<thead>
<tr>
<th>( N(x) )</th>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0, 0, 0]</td>
<td>0</td>
</tr>
<tr>
<td>[1, 0, 1, 0]</td>
<td>29</td>
</tr>
<tr>
<td>[1, 0, 0, 1]</td>
<td>32</td>
</tr>
</tbody>
</table>
Knapsack Problem

\[
\text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4
\]

Subject to

\[
2x_1 + 2x_2 + x_3 + x_4 \leq 3
\]

\[
x_1, x_2, x_3, x_4 = 0 \text{ or } 1.
\]

- Initial sol. \( x^0 = x = [1, 0, 0, 0] \)
  \[
f(x) = 18
\]

- \( N(x) \)

<table>
<thead>
<tr>
<th>N(x)</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0, 0, 0]</td>
<td>0</td>
</tr>
<tr>
<td>[1, 0, 1, 0]</td>
<td>29</td>
</tr>
<tr>
<td>[1, 0, 0, 1]</td>
<td>32</td>
</tr>
</tbody>
</table>

- \( x' = [1, 0, 0, 1] \); \( f(x') = 32 \)
- Since \( f(x') > f(x) \), then
  \( x' \) replaces the current sol. \( x \)
  and we start a new iteration
Knapsack Problem

\[ \max f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4 \]

Subject to
\[
2x_1 + 2x_2 + x_3 + x_4 \leq 3 \\
x_1, x_2, x_3, x_4 = 0 \text{ or } 1.
\]

- current sol. \( x = [1, 0, 0, 1] \); 
  \[ f(x) = 32 \]

<table>
<thead>
<tr>
<th>( N(x) )</th>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0, 0, 1]</td>
<td>14</td>
</tr>
<tr>
<td>[1, 0, 0, 0]</td>
<td>18</td>
</tr>
</tbody>
</table>
Knapsack Problem

\[ \max f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4 \]

Subject to
\[
2x_1 + 2x_2 + x_3 + x_4 \leq 3
\]
\[
x_1, x_2, x_3, x_4 = 0 \text{ or } 1.
\]

• Current sol. \( x = [1, 0, 0, 1] \);
  \[ f(x) = 32 \]

• \[
\begin{array}{c|c}
N(x) & f \\
[0, 0, 0, 1] & 14 \\
[1, 0, 0, 0] & 18 \\
\end{array}
\]

• \( x' = [1, 0, 0, 0] \); \( f(x') = 18 \)

• Since \( f(x') < f(x) \), then
  the procedure stops with
  \( x^* = x \)
Tabu Search

- Tabu Search is an iterative neighborhood or local search technique
- At each iteration we move from a current solution \( x \) to a new solution \( x' \) in a neighborhood of \( x \) denoted \( N(x) \),
- until we reach some solution \( x^* \) acceptable according to some criterion
Selecting $x'$
Selecting $x'$
Selecting $x'$
Selecting $x'$
Selecting $x'$
Selecting $x'$
Selecting $x'$
Tabu Search (TS)
Tabu Search ($TS$)

- **Initialize**
  Select an initial solution $x^0 \in X$

Let $x := x^0$
Tabu Search (TS)

- **Initialize**
  Select an initial solution $x^0 \in X$

  Let $x := x^0$ and $stop := false$

- **While not stop**

  Determine a subset $NC(x) \subseteq N(x)$

  Determine $x' \in NC(x)$ such that
  
  $x' := \arg\min_{z \in NC(x)} \{ f(z) \}$

  At each iteration, a best solution $x' \in NC(x)$ is selected.
Tabu Search (TS)

- **Initialize**
  Select an initial solution $x^0 \in X$

  Let $x := x^0$ and $stop := false$

- **While not stop**

  Determine a subset $NC(x) \subseteq N(x)$

  Determine $x' \in NC(x)$ such that
  $$x' := \arg\min_{z \in NC(x)} \{ f(z) \}$$

  $x := x'$

At each iteration, a best solution $x' \in NC(x)$ is selected.

$x' \in NC(x)$ is the current solution for the next iteration.
Tabu Search (TS)

- **Initialize**
  Select an initial solution $x^0 \in X$

  Let $x := x^0$ and $stop := false$

- **While** not $stop$

  Determine a subset $NC(x) \subseteq N(x)$

  Determine $x' \in NC(x)$ such that
  
  $$x' := \arg\min_{z \in NC(x)} \{ f(z) \}$$

  $x := x'$

- As long as $x'$ is better than $x$, the behavior of the procedure is similar to that of the descent method.

- Otherwise, moving to $x'$ as the next current solution induces no improvement or a deterioration of the objective function, but it allows to move out of a local minimum.
**Tabu Search (TS)**

- **Initialize**
  Select an initial solution $x^0 \in X$
  Let $TL_k = \emptyset$, $k = 1, 2, \ldots, p$

  Let $x := x^0$ and $stop := false$

- **While not stop**

  Determine a subset $NC(x) \subseteq N(x)$ of solutions
  $z = x \oplus m$ such that
  $t_k(m)$ is not in $TL_k$, $k = 1, 2, \ldots, p$

  Determine $x' \in NC(x)$ such that
  $x' := \arg\min_{z \in NC(x)} \{ f(z) \}$
  $x := x'$

- As long as $x'$ is better than $x$, the behavior of the procedure is similar to that of the descent method.

- Otherwise, moving to $x'$ as the next current solution induces no improvement or a deterioration of the objective function, but it allows to move out of a local minimum.

  To prevent cycling, recently visited solutions are eliminated from $NC(x)$ using Tabu lists.
Tabu Lists (TL)

• Short term Tabu lists $TL_k$ are used to remember attributes or characteristics of the modification used to generate the new current solution.

• A Tabu List often used for the assignment type problem is the following:

  If the new current solution $x'$ is generated from $x$ by modifying the resource of $i$ from $j(i)$ to $p$, then the pair $(i, j(i))$ is introduced in the Tabu list $TL$.

  If the pair $(i, j)$ is in $TL$, then any solution where resource $j$ is to be assigned to $i$ is declared Tabu.

• The Tabu lists are cyclic in order for an attribute to remain Tabu for a fixed number $n_k$ of iterations.
Tabu Search (TS)

- **Initialize**
  Select an initial solution $x^0 \in X$
  Let $TL_k = \emptyset$, $k = 1, 2, \ldots, p$

  Let $x := x^0$ and $stop := false$

- **While** not stop

  Determine a subset $NC(x) \subseteq N(x)$ of solutions
  $z = x \oplus m$ such that
  $t_k(m)$ is not in $TL_k$, $k = 1, 2, \ldots, p$

  Determine $x' \in NC(x)$ such that
  $x' := \arg\min_{z \in NC(x)} \{ f(z) \}$
  $x := x'$

  Update Tabu Lists $TL_k$, $k = 1, 2, \ldots, p$

- As long as $x'$ is better than $x$, the behavior of the procedure is similar to that of the descent method.

- Otherwise, moving to $x'$ as the next current solution induces no improvement or a deterioration of the objective function, but it allows to move out of a local minimum.

To prevent cycling, recently visited solutions are eliminated from $NC(x)$ using *Tabu lists*.
Tabu Search (TS)

• Initialize
  Select an initial solution $x^0 \in X$
  Let $TL_k = \Phi$, $k = 1, 2, \ldots, p$

  Let $x^* := x := x^0$ and $stop := false$

• While not stop

  Determine a subset $NC(x) \subseteq N(x)$ of solutions $z = x \oplus m$ such that
  $t_k(m)$ is not in $TL_k$, $k = 1, 2, \ldots, p$
  or
  $f(z) < f(x^*)$

  Determine $x' \in NC(x)$ such that
  $x' := \arg\min_{z \in NC(x)} \{ f(z) \}$
  $x := x'$

  Update Tabu Lists $TL_k$, $k = 1, 2, \ldots, p$

• Since Tabu lists are specified in terms of attributes of the modifications used, we required an **Aspiration criterion** to bypass the tabu status of good solutions declared Tabu without having been visited recently

  may include $z$ in $NC(x)$ even if $z$ is Tabu whenever $f(z) < f(x^*)$

  where $x^*$ is the best solution found so far
Tabu Search ($TS$)

- **Initialize**
  Select an initial solution $x^0 \in X$
  Let $TL_k = \emptyset$, $k = 1, 2, \ldots, p$

  Let $x^* := x := x^0$ and $stop := false$

- **While** $not$ $stop$

  Determine a subset $NC(x) \subseteq N(x)$ of solutions $z = x \oplus m$ such that
  
  - $t_k(m)$ is not in $TL_k$, $k = 1, 2, \ldots, p$
  - $or$
    - $f(z) < f(x^*)$

  Determine $x' \in NC(x)$ such that
  
  $x' := \arg\min_{z \in NC(x)} \{ f(z) \}$

  $x := x'$

  **If** $f(x) < f(x^*)$ **then** $x^* := x$

  Update Tabu Lists $TL_k$, $k = 1, 2, \ldots, p$
Tabu Search (TS)

- **Initialize**
  Select an initial solution $x^0 \in X$
  Let $T_{L_k} = \emptyset$, $k = 1, 2, \ldots, p$

  Let $x^* := x := x^0$ and $stop := false$

- **While** not $stop$

  Determine a subset $NC(x) \subseteq N(x)$ of solutions
  $z = x \oplus m$ such that
  $t_k(m)$ is not in $T_{L_k}$, $k = 1, 2, \ldots, p$
  or
  $f(z) < f(x^*)$

  Determine $x' \in NC(x)$ such that
  $x' := \text{argmin}_{z \in NC(x)} \{ f(z) \}$
  $x := x'$
  **If** $f(x) < f(x^*)$ **then** $x^* := x$

  Update Tabu Lists $T_{L_k}$, $k = 1, 2, \ldots, p$

- **No monotonicity of the objective function!!!**
- **Stopping criterion ???**
Tabu Search (TS)

- **Initialize**
  Select an initial solution \( x^0 \in X \)
  Let \( TL_k = \emptyset, k = 1, 2, ..., p \)
  Let \( \text{iter} := \text{niter} := 0 \)
  Let \( x^* := x := x^0 \) and \( \text{stop} := \text{false} \)

- **While not stop**
  \( \text{iter} := \text{iter} + 1 ; \text{niter} := \text{niter} + 1 \)
  Determine a subset \( NC(x) \subseteq N(x) \) of solutions \( z = x \oplus m \) such that
  \( t_k(m) \) is not in \( TL_k \), \( k = 1, 2, ..., p \)
  or \( f(z) < f(x^*) \)
  Determine \( x' \in NC(x) \) such that
  \( x' := \arg\min_{z \in NC(x)} \{ f(z) \} \)
  \( x := x' \)
  **If** \( f(x) < f(x^*) \) **then** \( x^* := x \), and \( \text{niter} := 0 \)
  **If** \( \text{iter} = \text{itermax} \) **or** \( \text{niter} = \text{nitermax} \)
  **then** \( \text{stop} := \text{true} \)
  Update Tabu Lists \( TL_k \), \( k = 1, 2, ..., p \)
- \( x^* \) is the best solution generated

Stopping criteria:
- maximum number of iterations
- maximum number of successive iterations where \( f(x^*) \) does not improve
Second example

Initial sol. $x^0 = x \Rightarrow [r, r, b, b] ; f(x) = 2$

\[
\begin{array}{c|c|c|c}
N(x) & f \\
\hline
TL = \Phi
\end{array}
\]
Second example

Initial sol. $x^0 = x \Rightarrow [r, r, b, b] ; f(x) = 2$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[b, r, b, b]</td>
<td>2</td>
</tr>
</tbody>
</table>
Second example

Initial sol. \( x^0 = x \Rightarrow [r, r, b, b] \); \( f(x) = 2 \)

<table>
<thead>
<tr>
<th>( N(x) )</th>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[b, r, b, b]</td>
<td>2</td>
</tr>
<tr>
<td>[r, b, b, b]</td>
<td>2</td>
</tr>
</tbody>
</table>
Second example

Initial sol. $x^0 = x \Rightarrow [r, r, b, b] ; f(x) = 2$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[b, r, b, b]</td>
<td>2</td>
</tr>
<tr>
<td>[r, b, b, b]</td>
<td>2</td>
</tr>
<tr>
<td>[r, r, r, b]</td>
<td>2</td>
</tr>
</tbody>
</table>
Second example

Initial sol. $x^0 = x \Rightarrow [r, r, b, b] ; f(x) = 2$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[b, r, b, b]</td>
<td>2</td>
</tr>
<tr>
<td>[r, b, b, b]</td>
<td>2</td>
</tr>
<tr>
<td>[r, r, r, b]</td>
<td>2</td>
</tr>
<tr>
<td>[r, r, b, r]</td>
<td>2</td>
</tr>
</tbody>
</table>
Second example

Initial sol. $x^0 = x \Rightarrow [r, r, b, b] ; f(x) = 2$  

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[b, r, b, b]</td>
<td>2</td>
</tr>
<tr>
<td>[r, b, b, b]</td>
<td>2</td>
</tr>
<tr>
<td>[r, r, r, b]</td>
<td>2</td>
</tr>
<tr>
<td>[r, r, b, r]</td>
<td>2</td>
</tr>
</tbody>
</table>

- Select randomly $x'$ in $N(x)$ (since all solutions have same value):
  
  $x' \Rightarrow [r, b, b, b]$  

- $x'$ replaces the current sol. $x$ and we start a new iteration  

$TL = \{ (2, r) \}$
Second example

Current sol. \( x \Rightarrow [r, b, b, b] ; f(x) = 2 \)

\[
\begin{array}{c|c}
N(x) & f \\
\hline
\end{array}
\]

\( TL = \{ (2, r) \} \)
Second example

Current sol. \( x \Rightarrow [r, b, b, b] ; f(x) = 2 \)

\[
\begin{array}{c|c}
N(x) & f \\
\hline
[b, b, b, b] & 4 \\
\end{array}
\]

\( TL = \{ (2, r) \} \)
Second example

Current sol. $x => [r, b, b, b] ; f(x) = 2$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
<th>TL</th>
</tr>
</thead>
<tbody>
<tr>
<td>[b, b, b, b]</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>[r, r, b, b]</td>
<td>2</td>
<td>tabu</td>
</tr>
</tbody>
</table>

$2N(x)$
Second example

Current sol. \( x \Rightarrow [r, b, b, b] ; f(x) = 2 \)

<table>
<thead>
<tr>
<th>( N(x) )</th>
<th>( f )</th>
<th>( TL = { (2, r) } )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[b, b, b, b]</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>[r, r, b, b]</td>
<td>2 ( tabu )</td>
<td></td>
</tr>
<tr>
<td>[r, b, r, b]</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Second example

Current sol. $x \Rightarrow [\text{r, b, b, b}] ; f(x) = 2$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
<th>$TL = {(2, \text{r})}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[b, b, b, b]</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>[r, r, b, b]</td>
<td>2</td>
<td>tabu</td>
</tr>
<tr>
<td>[r, b, r, b]</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>[r, b, b, r]</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Second example

Current sol. \( x \Rightarrow [r, b, b, b] ; f(x) = 2 \)

\[
\begin{array}{c|c|c}
N(x) & f & TL \\
\hline
[b, b, b, b] & 4 & \text{tabu} \\
[r, r, b, b] & 2 & \\
[r, b, r, b] & 0 & \\
[r, b, b, r] & 2 & \\
\end{array}
\]

- \( x' \Rightarrow [r, b, r, b] ; f(x') = 0 \)
- Since \( f(x') = 0 \) then
  the procedure stops with
\[
x^* = x' \Rightarrow [r, b, r, b]
\]
Knapsack Problem

• Problem formulation:
  \[ \text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4 \]
  Subject to
  \[ 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \]
  \[ x_1, x_2, x_3, x_4 = 0 \text{ or } 1. \]

• Neighborhood \( N(x) \) specified by the following modification or move:

  The value of one and only one variable is modified (from 0 to 1 or from 1 to 0) to generate a new solution satisfying constraint
  \[ 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \]

• Tabu list \( TL \) :
  element introduced in the Tabu list is the pair
  (index of modified variable, current value of the modified variable)

• Length of Tabu list = 2

• \( itermax = 5 \)
  \( nitermax = 4 \)
Knapsack Problem

$\text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4$

Subject to
\[
2x_1 + 2x_2 + x_3 + x_4 \leq 3
\]
\[
x_1, x_2, x_3, x_4 = 0 \text{ or } 1.
\]

- Initial sol. $x^0 = x = [1, 0, 0, 0]$ ;
  $f(x) = 18$

- $TL = \Phi$

- $iter = 1$
  $niter = 1$

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0, 0, 0]</td>
<td>0</td>
</tr>
<tr>
<td>[1, 0, 1, 0]</td>
<td>29</td>
</tr>
<tr>
<td>[1, 0, 0, 1]</td>
<td>32</td>
</tr>
</tbody>
</table>
Knapsack Problem

\[ \text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4 \]

Subject to
\[ 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \]
\[ x_1, x_2, x_3, x_4 = 0 \text{ or } 1. \]

- Initial sol. \( x^0 = x = x^* = [1, 0, 0, 0] \);
  \[ f(x) = 18 \]

- \( N(x) \)  \begin{tabular}{c|c}
  \hline
  \( [0, 0, 0, 0] \) & 0 \\
  \( [1, 0, 1, 0] \) & 29 \\
  \( [1, 0, 0, 1] \) & 32 \\
  \hline
\end{tabular}

- \( x' = [1, 0, 0, 1] ; f(x') = 32 \)
- Since \( f(x') > f(x^*) = 18 \), then \( x^* = x' \), and \( niter := 0 \)
- \( x' \) replaces the current sol. \( x \) and we start a new iteration
- \( TL = \{ (4, 0) \} \)

- Tabu list = \( \Phi \)

- \( \text{iter} = 1 \)

- \( niter = 1 \)
Knapsack Problem

\[
max f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4
\]

Subject to
\[
2x_1 + 2x_2 + x_3 + x_4 \leq 3
\]
\[
x_1, x_2, x_3, x_4 = 0 \text{ or } 1.
\]

- Current sol. \( x = [1, 0, 0, 1] \);
  \( f(x) = 32 \)

- \( N(x) \)           \( f \)  
  \[
  \begin{array}{l|c}
  [0, 0, 0, 1] & 14 \\
  [1, 0, 0, 0] & 18 <= tabu
  \end{array}
  \]

- \( TL = \{ (4, 0) \} \)

- \( iter = 2 \)
  \( niter = 1 \)
Knapsack Problem

\[
\text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4
\]

Subject to

\[
2x_1 + 2x_2 + x_3 + x_4 \leq 3
\]
\[
x_1 , x_2 , x_3 , x_4 = 0 \text{ or } 1.
\]

• Current sol. \( x = [1, 0, 0, 1] \);
  \[
  f(x) = 32
  \]

\[
\begin{array}{|c|c|}
\hline
N(x) & f \\
\hline
[0, 0, 0, 1] & 14 \\
[1, 0, 0, 0] & 18 \leq \text{tabu} \\
\hline
\end{array}
\]

• \( x' = [0, 0, 0, 1] ; f(x') = 14 \)
• Since \( f(x') < f(x^*) = 32 \), then \( x^* \) not modified
• \( x' \) replaces the current sol. \( x \) and we start a new iteration
• \( TL = \{ (4, 0), (1, 1) \} \)

• \( TL = \{ (4, 0) \} \)
• \( \text{iter} = 2 \)
  \[
  niter = 1
  \]
Knapsack Problem

\[ \text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4 \]

Subject to
\[
2x_1 + 2x_2 + x_3 + x_4 \leq 3
\]
\[
x_1, x_2, x_3, x_4 = 0 \text{ or } 1.
\]

- Current sol. \( x = [0, 0, 0, 1] \);
  \[ f(x) = 14 \]

- \( TL = \{ (4, 0), (1, 1) \} \)

- \( \text{iter} = 3 \)
  \[ niter = 2 \]

- \( \begin{array}{c|c}
N(x) & f \\
\hline
[1, 0, 0, 1] & 32 \leq \text{tabu} \\
[0, 1, 0, 1] & 39 \\
[0, 0, 1, 1] & 25 \\
[0, 0, 0, 0] & 0 \leq \text{tabu}
\end{array} \)
Knapsack Problem

\[ \text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4 \]

Subject to
\[ 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \]
\[ x_1, x_2, x_3, x_4 = 0 \text{ or } 1. \]

- Current sol. \( x = [0, 0, 0, 1] \);
  \[ f(x) = 14 \]
- \[ N(x) \]
  \[ \begin{array}{c|c}
  N(x) & f \\
  \hline
  [1, 0, 0, 1] & 32 \quad \text{\( \leq \)} \text{ tabu} \\
  [0, 1, 0, 1] & 39 \\
  [0, 0, 1, 1] & 25 \\
  [0, 0, 0, 0] & 0 \quad \text{\( \leq \)} \text{ tabu} \\
  \end{array} \]
- \( x' = [0, 1, 0, 1] \); \( f(x') = 39 \)
- Since \( f(x') > f(x^*) = 32 \), then
  \[ x^* = x', \text{ and } niter := 0 \]
- \( x' \) replaces the current sol. \( x \) and we start a new iteration
- \( TL = \{(1, 1), (2, 0)\} \)
Knapsack Problem

\[
\text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4
\]

Subject to
\[
2x_1 + 2x_2 + x_3 + x_4 \leq 3
\]
\[
x_1, x_2, x_3, x_4 = 0 \text{ or } 1.
\]

- Current sol. \( x = [0, 1, 0, 1] \);
  \[
  f(x) = 39
  \]

- \( N(x) \)

<table>
<thead>
<tr>
<th>( N(x) )</th>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0, 0, 1]</td>
<td>14 &amp;&amp; ( \leq \text{tabu} )</td>
</tr>
<tr>
<td>[0, 1, 0, 0]</td>
<td>25</td>
</tr>
</tbody>
</table>

- \( TL = \{ (1, 1), (2, 0) \} \)

- \( \text{iter} = 4 \)
  \( niter = 1 \)
Knapsack Problem

\[
\text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4
\]

Subject to
\[
\begin{align*}
2x_1 + 2x_2 + x_3 + x_4 & \leq 3 \\
x_1, x_2, x_3, x_4 & = 0 \text{ or } 1.
\end{align*}
\]

- Current sol. \(x = [0, 1, 0, 1]\);
  \[f(x) = 39\]

- \(N(x)\)

<table>
<thead>
<tr>
<th>(f)</th>
<th>(N(x))</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>[0, 0, 0, 1]</td>
</tr>
<tr>
<td>25</td>
<td>[0, 1, 0, 0]</td>
</tr>
</tbody>
</table>

- Since \(f(x') < f(x*) = 39\), then \(x^*\) is not modified
- \(x' = [0, 1, 0, 0]\); \(f(x') = 25\)
- \(TL = \{(1, 1), (2, 0)\}\)
- \(niter = 1\)
- \(iter = 4\)
- \(\text{TL} = \{(2, 0), (4, 1)\}\)
Knapsack Problem

\[ \text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4 \]

Subject to

\[ 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \]
\[ x_1, x_2, x_3, x_4 = 0 \text{ or } 1. \]

- \( TL = \{ (2, 0), (4, 1) \} \)
- \( \text{iter} = 5 \)
- \( \text{niter} = 2 \)
- Stop since \( \text{iter} = \text{itermax} = 5 \)
- The solution
  \[ x^* = [0, 1, 0, 1] \]
  \[ f(x^*) = 39 \]
Simulated Annealing

- Probabilistic technique allowing to move out of local minima

- At each iteration, a solution $x'$ is selected randomly in a subset of the neighborhood $N(x)$

- This approach was already used to simulate the evolution of an unstable physical system toward a thermodynamic stable equilibrium point at a fixed temperature
Simulated Annealing

- **Initialize**
  Select an initial solution $x^0 \in X$

Let $x := x^0$
Simulated Annealing

- **Initialize**
  Select an initial solution $x^0 \in X$

  Let $x := x^0$ and $stop := false$

- **While** not $stop$

  Generate randomly $x' \in V \subseteq N(x)$

  At each iteration a solution $x'$ in a subset $V \subseteq N(x)$ is selected randomly
Simulated Annealing

- **Initialize**
  Select an initial solution $x^0 \in X$

  Let $x := x^0$ and $stop := false$

- **While** not $stop$
  
  Generate randomly $x' \in V \subseteq N(x)$
  
  If $\Delta f = f(x') - f(x) < 0$ then $x := x'$

As long as $x'$ is better than $x$ ($\Delta f < 0$), the behavior of the procedure is similar to that of the descent method.
Simulated Annealing

- **Initialize**
  Select an initial solution $x^0 \in X$

  Let $x := x^0$ and $stop := false$

- **While** not stop

  Generate randomly $x' \in V \subseteq N(x)$
  
  If $\Delta f = f(x') - f(x) < 0$ then $x := x'$

  else

  Generate random number $r \in (0, 1]$
  
  If $r < e^{-\Delta f / T}$ then $x := x'$

  if $\Delta f \geq 0$, the probability that $x'$ replaces $x$ decreases as $\Delta f$ increases and as the number of iterations already completed increases
Simulated Annealing

- **Initialize**
  Select an initial solution \( x^0 \in X \)
  Select an initial temperature \( TP^0 \)
  Let \( TP := TP^0 \)
  Let \( x := x^0 \) and \( stop := false \)

- **While** \( not \ stop \)

  Generate randomly \( x' \in V \subseteq N(x) \)
  \textbf{If} \( \Delta f = f(x') - f(x) < 0 \) \textbf{then} \( x := x' \)
  \textbf{else}
  Generate random number \( r \in (0, 1] \)
  \textbf{If} \( r < e^{-\Delta f/TP} \) \textbf{then} \( x := x' \)

  \( TP := \alpha TP \)

if \( \Delta f \geq 0 \), the probability that \( x' \) replaces \( x \) decreases as \( \Delta f \) increases and as the number of iterations already completed increases.

A temperature factor \( TP \) is used to account for the number of iterations increasing. Thus \( TP \) is multiplied by a parameter value \( \alpha \in (0, 1) \) in order for \( TP \) to decrease with the number of iterations.
Simulated Annealing

- **Initialize**
  Select an initial solution \( x^0 \in X \)
  Select an initial temperature \( TP^0 \)
  Let \( TP := TP^0 \)
  Let \( x := x^0 \) and \( stop := false \)

- **While** not stop
  
  \[ minoriter := 0 \]
  
  **While** \( minoriter \leq minoritermax \)
  
  \[ minoriter := minoriter + 1 \]
  
  Generate randomly \( x' \in V \subseteq N(x) \)
  
  **If** \( \Delta f = f(x') - f(x) < 0 \) **then** \( x := x' \)
  
  **else**
  
  Generate random number \( r \in (0, 1] \)
  
  **If** \( r < e^{-\Delta f/TP} \) **then** \( x := x' \)

  \( TP := \alpha TP \)

- \( minoritermax \) iterations are completed using the same temperature \( TP \)

- if \( \Delta f \geq 0 \), the probability that \( x' \) replaces \( x \) decreases as \( \Delta f \) increases and as the number of iterations already completed increases

- A temperature factor \( TP \) is used to account for the number of iteration increasing. Thus \( TP \) is multiplied by a parameter value \( \alpha \in (0, 1) \) in order for \( TP \) to decrease with the number of iterations by multiplying
Simulated Annealing

- **Initialize**
  Select an initial solution \( x^0 \in X \)
  Select an initial temperature \( T^0 \)
  Let \( T^P := T^0 \)
  Let \( x^* := x := x^0 \) and \( stop := false \)

- **While not stop**

  \[ \text{minoriter} := 0 \]
  
  **While** \( \text{minoriter} \leq \text{minoritermax} \)
  
  \[ \text{minoriter} := \text{minoriter} + 1 \]
  
  Generate randomly \( x' \in V \subseteq N(x) \)
  
  If \( \Delta f = f(x') - f(x) < 0 \) then \( x := x' \)
  
  else

  Generate random number \( r \in (0, 1] \)
  
  If \( r < e^{-\frac{\Delta f}{TP}} \) then \( x := x' \)

  If \( f(x) < f(x^*) \) then \( x^* := x \)

  \( TP := \alpha \cdot TP \)

  Update \( x^* \) the best solution found so far
Simulated Annealing

- **Initialize**
  Select an initial solution $x^0 \in X$
  Select an initial temperature $T_P^0$
  Let $T_P := T_P^0$
  Let $x^* := x := x^0$ and stop := false

- **While not stop**

  $\text{minoriter} := 0$
  **While** $\text{minoriter} \leq \text{minoritermax}$
     $\text{minoriter} := \text{minoriter} + 1$
     Generate randomly $x' \in V \subseteq N(x)$
     \textbf{If} $\Delta f = f(x') - f(x) < 0$ \textbf{then} $x := x'$
     \textbf{else}
        Generate random number $r \in (0, 1]$
        \textbf{If} $r < e^{-\Delta f / T_P}$ \textbf{then} $x := x'$
        \textbf{If} $f(x) < f(x^*)$ \textbf{then} $x^* := x$

  $T_P := \alpha T_P$

- No monotonicity of the objective function!!!
- Stopping criterion ???
Simulated Annealing

- **Initialize**
  Select an initial solution \( x^0 \in X \)
  Select an initial temperature \( TP^0 \)
  Let \( iter := niter := 0 \) and \( TP := TP^0 \)
  Let \( x^* := x := x^0 \) and \( stop := false \)

- **While not stop**
  \( iter := iter + 1 \); \( niter := niter + 1 \)
  \( minoriter := 0 \); \( Value := f(x) \)
  **While** \( minoriter \leq minoritermax \)
    \( minoriter := minoriter + 1 \)
    Generate randomly \( x' \in V \subseteq N(x) \)
    \( \text{If} \ \Delta f = f(x') - f(x) < 0 \ \text{then} \ x := x' \)
    \( \text{else} \)
    Generate random number \( r \in (0, 1] \)
    \( \text{If} \ r < e^{-\Delta f/TP} \ \text{then} \ x := x' \)
    \( \text{If} \ f(x) < f(x^*) \ \text{then} \ x^* := x \), and \( niter := 0 \)
  \( \text{If} \ f(x) < Value \ \text{then} \ niter := 0 \)
  \( TP := \alpha TP \)
  **If** \( iter = itermax \) \( \text{or} \ niter = nitermax \)
  **then** \( stop := true \)

- **Stopping criteria:**
  - maximum number of different temperature values
  - maximum number of successive different temperature values where \( f(x) \) does not improve

- \( x^* \) is the best solution generated
Knapsack Problem

- Problem formulation:

\[ \max f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4 \]
Subject to
\[ 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \]
\[ x_1, x_2, x_3, x_4 = 0 \text{ or } 1. \]

- Neighborhood \( N(x) \) specified by the following modification or move:

The value of one and only one variable is modified (from 0 to 1 or from 1 to 0) to generate a new solution satisfying constraint

\[ 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \]

- \( TP^0 = 20 \)

- \( \alpha = 0.5 \)

- \( \text{itermax} = 3 \)
  \[ \text{minoritermax} = 2 \]
  \[ \text{nitermax} = 2 \]

- When random decisions are required, use the following sequence of random numbers:
  0.72, 0.83, 0.55, 0.41, 0.09, 0.64
Knapsack Problem

$max \, f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4$

Subject to

\[ 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \]
\[ x_1, x_2, x_3, x_4 = 0 \text{ or } 1. \]

Random numbers sequence:

0.72, 0.83, 0.55, 0.41, 0.09, 0.64

- Initial sol. $x^0 = x = [1, 0, 0, 0]$ ; $f(x) = 18$ ; Value = 18

<table>
<thead>
<tr>
<th>$N(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0, 0, 0]</td>
<td>0</td>
</tr>
<tr>
<td>[1, 0, 1, 0]</td>
<td>29</td>
</tr>
<tr>
<td>[1, 0, 0, 1]</td>
<td>32</td>
</tr>
</tbody>
</table>

- $TP = 20$
- $iter = 1$
  
  $minoriter = 1$
  
  $niter = 1$
Knapsack Problem

\[ \text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4 \]

Subject to
\[ 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \]
\[ x_1, x_2, x_3, x_4 = 0 \text{ or } 1. \]

Random numbers sequence:
\[ 0.72, 0.83, 0.55, 0.41, 0.09, 0.64 \]

- Initial sol. \( x^0 = x = [1, 0, 0, 0] \);
  \[ f(x) = 18 \text{ ; Value } = 18 \]

<table>
<thead>
<tr>
<th>( N(x) )</th>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0, 0, 0]</td>
<td>0</td>
</tr>
<tr>
<td>[1, 0, 1, 0]</td>
<td>29</td>
</tr>
<tr>
<td>[1, 0, 0, 1]</td>
<td>32</td>
</tr>
</tbody>
</table>

- \( TP = 20 \)
- \( \text{iter} = 1 \)
- \( \text{minoriter} = 1 \)
- \( niter = 1 \)

\[ x' = [1, 0, 0, 1] \text{ ; } f(x') = 32 \]
Knapsack Problem

\[ \text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4 \]

Subject to
\[ 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \]
\[ x_1, x_2, x_3, x_4 = 0 \text{ or } 1. \]

Random numbers sequence:
0.72, 0.83, 0.55, 0.41, 0.09, 0.64

- \text{TP} = 20
- \text{iter} = 1
  - \text{minoriter} = 1
  - \text{niter} = 1

- Initial sol. \( x^0 = x = [1, 0, 0, 0] \); \( f(x) = 18 \); \text{Value} = 18

<table>
<thead>
<tr>
<th>( N(x) )</th>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0, 0, 0]</td>
<td>0</td>
</tr>
<tr>
<td>[1, 0, 1, 0]</td>
<td>29</td>
</tr>
<tr>
<td>[1, 0, 0, 1]</td>
<td>32</td>
</tr>
</tbody>
</table>

\[ x' = [1, 0, 0, 1] \]; \( f(x') = 32 \)

- Since \( f(x') > f(x) = 18 \), then \( x' \) replaces the current sol. \( x \).
Knapsack Problem

\[
\max f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4
\]

Subject to
\[
2x_1 + 2x_2 + x_3 + x_4 \leq 3
\]
\[
x_1, x_2, x_3, x_4 = 0 \text{ or } 1.
\]

Random numbers sequence:
0.72, 0.83, 0.55, 0.41, 0.09, 0.64

- Sol. \( x = [1, 0, 0, 1] \);
  \[ f(x) = 32 \quad ; \quad \text{Value} = 18 \]

- \[
\begin{array}{|c|c|}
\hline
N(x) & f \\
\hline
[0, 0, 0, 1] & 14 \\
[1, 0, 0, 0] & 18 \\
\hline
\end{array}
\]

- \( TP = 20 \)
- \( iter = 1 \)
  \( minoriter = 2 \)
  \( niter = 1 \)
Knapsack Problem

\[ \text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4 \]

Subject to
\[ 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \]
\[ x_1, x_2, x_3, x_4 = 0 \text{ or } 1. \]

Random numbers sequence:
0.72, 0.83, 0.55, 0.41, 0.09, 0.64

- Sol. \( x = [1, 0, 0, 1] \);
  \[ f(x) = 32 \text{ ; } \text{Value} = 18 \]
- \( N(x) \) \[ \begin{array}{c|c}
  x & f \\
  \hline
  [0, 0, 0, 1] & 14 \\
  [1, 0, 0, 0] & 18 \\
\end{array} \]

- \( TP = 20 \)
- \( \text{iter} = 1 \)
- \( \text{minoriter} = 2 \)
- \( niter = 1 \)

\[ x' = [1, 0, 0, 0] \text{ ; } f(x') = 18 \]
Knapsack Problem

\[ \text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4 \]

Subject to
\[ 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \]
\[ x_1, x_2, x_3, x_4 = 0 \text{ or } 1. \]

Random numbers sequence:
0.72, 0.83, 0.55, 0.41, 0.09, 0.64

- **TP** = 20
- **iter** = 1 *minoriter = 2* *niter = 1*

- Sol. \( x = [1, 0, 0, 1] \);
  \[ f(x) = 32 \quad ; \quad \text{Value} = 18 \]

<table>
<thead>
<tr>
<th>( N(x) )</th>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0, 0, 1]</td>
<td>14</td>
</tr>
<tr>
<td>[1, 0, 0, 0]</td>
<td>18</td>
</tr>
<tr>
<td>[1, 0, 0, 0]</td>
<td>18</td>
</tr>
</tbody>
</table>

\[ x' = [1, 0, 0, 0] ; \quad f(x') = 18 \]

\[ \Delta f = 18 - 32 = -14 \]

- Since \( e^{-14/20} \approx 0.497 < 0.55 \)
  we keep the same current solution \( x \)
Knapsack Problem

\[ \text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4 \]

Subject to
\[ 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \]
\[ x_1, x_2, x_3, x_4 = 0 \text{ or } 1. \]

Random numbers sequence:
0.72, 0.83, 0.55, 0.41, 0.09, 0.64

- Sol. \( x = [1, 0, 0, 1] \);
  \[ f(x) = 32 \quad ; \quad \text{Value} = 32 \]

- \[ \begin{array}{c|c}
N(x) & f \\
[0, 0, 0, 1] & 14 \\
[1, 0, 0, 0] & 18 \\
\end{array} \]

- \( TP = \alpha TP = 0.5*20 = 10 \)
- \( \text{iter} = 2 \)
  \[ \text{minoriter} = 1 \]
  \[ \text{niter} = 1 \]
Knapsack Problem

\[
\text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4
\]

Subject to

\[
\begin{align*}
2x_1 + 2x_2 + x_3 + x_4 & \leq 3 \\
x_1, x_2, x_3, x_4 & = 0 \text{ or } 1.
\end{align*}
\]

Random numbers sequence:

0.72, 0.83, 0.55, 0.41, 0.09, 0.64

- \( TP = 10 \)
- \( \text{iter} = 2 \)
- \( \text{minoriter} = 1 \)
- \( niter = 1 \)

\[
\text{Sol. } x = [1, 0, 0, 1] ; \quad f(x) = 32 \quad ; \quad \text{Value} = 32
\]

\[
\begin{array}{|c|c|}
\hline
N(x) & f \\
\hline
[0, 0, 0, 1] & 14 \\
[1, 0, 0, 0] & 18 \\
\hline
\end{array}
\]

\[
x' = [0, 0, 0, 1] ; \quad f(x') = 14
\]
Knapsack Problem

\[ \text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4 \]

Subject to
\[ 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \]
\[ x_1, x_2, x_3, x_4 = 0 \text{ or } 1. \]

Random numbers sequence:
0.72, 0.83, 0.55, 0.41, 0.09, 0.64

- \( TP = 10 \)
- \( \text{iter} = 2 \)

- \( \text{minoriter} = 1 \)
- \( niter = 1 \)

Sol. \( x = [1, 0, 0, 1] ; \)
\[ f(x) = 32 \ ; \ Value = 32 \]

\[
\begin{array}{c|c}
N(x) & f \\
\hline
[0, 0, 0, 1] & 14 \\
[1, 0, 0, 0] & 18 \\
\end{array}
\]

\[ x' = [0, 0, 0, 1] \ ; \ f(x') = 14 \]
\[ \Delta f = 14 - 32 = - 18 \]

Since \( e^{-18/10} \approx 0.165 > 0.09 \), then \( x' \) replaces the current sol. \( x \)
Knapsack Problem

\[ \text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4 \]

Subject to
\[ 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \]
\[ x_1, x_2, x_3, x_4 = 0 \text{ or } 1. \]

Random numbers sequence:
0.72, 0.83, 0.55, 0.41, 0.09, 0.64

- Sol. \( x = [0, 0, 0, 1] \);
  \[ f(x) = 14 ; \text{ Value } = 32 \]

- \( T_P = 10 \)
- \( \text{iter} = 2 \)
  \[ \text{minoriter} = 2 \]
  \[ \text{niter} = 1 \]

\[
\begin{array}{|c|c|}
\hline
N(x) & f \\
\hline
[1, 0, 0, 1] & 32 \\
[0, 1, 0, 1] & 39 \\
[0, 0, 1, 1] & 25 \\
[0, 0, 0, 0] & 0 \\
\hline
\end{array}
\]
Knapsack Problem

\[ \text{max } f(x) = 18x_1 + 25x_2 + 11x_3 + 14x_4 \]

Subject to
\[ 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \]
\[ x_1, x_2, x_3, x_4 = 0 \text{ or } 1. \]

Random numbers sequence:
0.72, 0.83, 0.55, 0.41, 0.09, 0.64

- \( TP = 10 \)
- \( \text{iter} = 2 \)
- \( \text{minoriter} = 2 \)
- \( \text{niter} = 1 \)

- Sol. \( x = [0, 0, 0, 1] ; \)
  \[ f(x) = 14 ; \text{Value} = 32 \]

<table>
<thead>
<tr>
<th>( N(x) )</th>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1, 0, 0, 1]</td>
<td>32</td>
</tr>
<tr>
<td>[0, 1, 0, 1]</td>
<td>39</td>
</tr>
<tr>
<td>[0, 0, 1, 1]</td>
<td>25</td>
</tr>
<tr>
<td>[0, 0, 0, 0]</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ x' = [0, 0, 1, 1] ; f(x') = 25 \]
- Since \( f(x') > f(x) = 14 \), then
  \( x' \) replaces the current sol. \( x \)
Knapsack Problem

\[
\begin{align*}
max \ f(x) &= 18x_1 + 25x_2 + 11x_3 + 14x_4 \\
\text{Subject to} \quad &2x_1 + 2x_2 + x_3 + x_4 \leq 3 \\
&x_1, x_2, x_3, x_4 = 0 \text{ or } 1.
\end{align*}
\]

Random numbers sequence:
0.72, 0.83, 0.55, 0.41, 0.09, 0.64

- \( TP = 5 \)
- \( \text{iter} = 3 \)
- \( \text{minoriter} = 0 \)
- \( niter = 2 \)

Sol. \( x = [0, 0, 1, 1] ; \)
\[
f(x) = 25 ; \text{Value} = 32
\]

Stop since \( \text{iter} = \text{itermax} = 3 \)

The solution
\[
x^* = [1, 0, 0, 1] \\
f(x^*) = 32
\]
Threshold Accepting

- Deterministic variant of Simulated Annealing:

- At each iteration a solution $x' \in V \subseteq N(x)$ is selected randomly.

- Solution $x'$ replaces $x$ if $\gamma(x', x) < dr$

  where $\gamma(x', x)$ is an auxiliary function and $dr$ is a threshold value.

- Stopping criteria:
  - maximum number of major iterations
  - maximum number of successive major iterations where $f(x)$ does not improve.
Simulated Annealing vs Threshold Accepting

- **Initialize**
  Select an initial solution $x^0 \in X$
  Select an initial temperature $TP^0$
  Let $\text{iter} := niter := 0$ and $TP := TP^0$
  Let $x^* := x := x^0$ and $\text{stop} := \text{false}$

- **While not stop**
  $\text{iter} := \text{iter} + 1$ ; $niter := niter + 1$
  minoriter := 0 ; $\text{Value} := f(x)$
  **While** minoriter < minoritermax
    minoriter := minoriter + 1
    Generate randomly $x' \in V \subseteq N(x)$
    **If** $\Delta f = f(x') - f(x) < 0$ **then** $x := x'$
    **else**
    Generate random number $r \in (0, 1]$ \\
    **If** $r < e^{-\Delta f / TP}$ **then** $x := x'$
  If $f(x) < f(x^*)$ **then** $x^* := x$ , and $niter := 0$
  If $f(x) < \text{Value}$ **then** $niter := 0$
  $TP := \alpha TP$
  **If** $\text{iter} = \text{itermax}$ or $niter = \text{nitermax}$
  **then** $\text{stop} := \text{true}$

- $x^*$ is the best solution generated

- **Initialize**
  Select an initial solution $x^0 \in X$
  Select an initial threshold value $dr^0$
  Let $\text{iter} := niter := 0$ and $dr := dr^0$
  Let $x^* := x := x^0$ and $\text{stop} := \text{false}$

- **While not stop**
  $\text{iter} := \text{iter} + 1$ ; $niter := niter + 1$
  minoriter := 0 ; $\text{Value} := f(x)$
  **While** minoriter < minoritermax
    minoriter := minoriter + 1
    Generate randomly $x' \in V \subseteq N(x)$
    **If** $\gamma(x', x) < dr$ **then** $x := x'$
    **else**
    **If** $f(x) < f(x^*)$ **then** $x^* := x$ , and $niter := 0$
  If $f(x) < \text{Value}$ **then** $niter := 0$
  Modify the threshold value $dr$
  **If** $\text{iter} = \text{itermax}$ or $niter = \text{nitermax}$
  **then** $\text{stop} := \text{true}$

- $x^*$ is the best solution generated
Standard Threshold Accepting (STA)

- Solution $x'$ replaces $x$
  
  if $\gamma(x', x) < dr$
  
  where $\gamma(x', x)$ is an auxiliary function and $dr$ is a threshold value

- $\gamma(x', x) = f(x') - f(x)$

- The threshold value $dr$ is updated as follows:
  
  $dr := b \cdot dr$
  
  where $b \in (0, 1)$ is a parameter
Great Deluge Method (GD)

- Solution $x'$ replaces $x$
  
  if $\gamma(x', x) < dr$

  where $\gamma(x', x)$ is an auxiliary function and $dr$ is a threshold value

- $\gamma(x', x) = f(x')$

- The initial threshold value $dr^0 = f(x^0)$,
  
  and the threshold value $dr$ is updated as follows:
  
  $dr := dr - \delta$

  where $\delta$ is a parameter
Maximal Deterioration Method ($MD$)

- Solution $x'$ replaces $x$
  
  $$\text{if } \gamma(x', x) < dr$$
  
  where $\gamma(x', x)$ is an auxiliary function and $dr$ is a threshold value

- $\gamma(x', x) = f(x')$

- The threshold value $dr$ is updated as follows:
  
  $$dr := f(x^*) + \mu$$
  
  where $\mu > 0$ is a parameter, and $x^*$ denotes the best solution found so far
Improving Strategies

• Intensification

• Multistart diversification strategies:
  - Random Diversification (RD)
  - First Order Diversification (FOD)

• Variable Neighborhood Search (VNS)

• Exchange Procedure
Intensification

• Intensification strategy used to search more extensively a promising region

• Two different ways (among others) of implementing:
  - Temporarily enlarge the neighborhood whenever the current solution induces a substantial improvement over the previous best known solution
  - Return to the best known solution to restart the NST using a temporarily enlarged neighborhood or using temporarily shorter Tabu lists
Diversification

• The diversification principle is complementary to the intensification. Its objective is to search more extensively the feasible domain by leading the NST to unexplored regions of the feasible domain.

• Numerical experiences indicate that it seems better to apply a short NST (of shorter duration) several times using different initial solutions rather than a long NST (of longer duration).
Random Diversification ($RD$)

Multistart procedure using new initial solutions generated randomly (with GRASP for instance)
First Order Diversification (FOD)

- Multistart procedure using the current local minimum $x^*$ to generate a new initial solution

- Move away from $x^*$ by modifying the current resources of some activities in order to generate a new initial solution in such a way as to deteriorate the value of $f$ as little as possible or even improve it, if possible
Variable Neighborhood Search (VNS)

- Specify (a priori) a set of *neighborhood structures* $N^k$, $k = 1, 2, \ldots, K$
Variable Neighborhood Search (VNS)

• Specify (a priori) a set of neighborhood structures $N^k$, $k = 1, 2, \ldots, K$

• At each major iteration, a “local minimum” $x''$ is generated using some (NST) where the initial solution $x'$ is selected randomly in $N^k(x^*)$, and using the neighborhood structure $N^k$
Variable Neighborhood Search (VNS)

- Specify (a priori) a set of neighborhood structures $N^k$, $k = 1, 2, \ldots, K$

- At each major iteration, a “local minimum” $x''$ is generated using some (NST) where the initial solution $x'$ is selected randomly in $N^k(x^*)$, and using the neighborhood structure $N^k$

  \textbf{if} $f(x'') < f(x^*) \textbf{ then}$ \quad $x''$ replaces $x^*$ and the neighborhood structure $N^l$ is used for the next major iteration

\begin{center}
Justification: we assume that it is easier to apply the NST with neighborhood structure $N^l$
\end{center}
Variable Neighborhood Search (VNS)

• Specify (a priori) a set of \textit{neighborhood structures} $N^k$, $k = 1, 2, \ldots, K$

• At each major iteration,
a “local minimum” $x''$ is generated using some (NST) where the initial solution $x'$ is selected randomly in $N^k(x^*)$, and using the neighborhood structure $N^k$

\textbf{if} $f(x'') < f(x^*)$ \textbf{then} \quad $x''$ replaces $x^*$ and
\hspace{1cm} the neighborhood structure $N^l$ is used for the next major iteration

\textbf{else} \quad the neighborhood structure $N^{k+l}$ is used for the next major iteration
Recall: Neighborhood for assignment type problem

• For the assignment type problem:
  Let $x$ be as follows: for each $1 \leq i \leq n$,
  
  $x_{ij(i)} = 1$
  
  $x_{ij} = 0$ for all other $j$

Each solution $x' \in N(x)$ is obtained by selecting an activity $i$ and modifying its resource from $j(i)$ to some other $p$ (i.e., the modification can be denoted $m = [i, p]$):

$x'_{ij(i)} = 0$

$x'_{ip} = 1$

$x'_{ij} = x_{ij}$ for all other $i, j$
Different neighborhood structures

For the assignment type problem, different neighborhood structures $N^k$ can be specified as follows:

Each solution $x' \in N^k(x)$ is obtained by selecting $k$ or less different activities $i$ and modifying their resource from $j(i)$ to some other $p_i$. 
Exchange Procedure

- Variant of (VNS) with 2 neighborhood structures where the descent method is the (NST) used:
  
i) Apply the descent method with neighborhood structure $N(x)$

ii) Once a local minimum $x^*$ is reached, apply the descent method using an enlarged and more complex neighborhood $EN(x)$ to find a new solution $x'$ such that $f(x') < f(x^*)$.

One way is to apply a truncated depth-first enumeration tree search where $x^*$ is the root. The search is truncated to monitor the number of modifications used (i.e., the depth of the tree or the size of the neighborhood) and the level of deterioration tolerated before reaching $x'$. 
Material from the following reference


ferland@iro.umontreal.ca