A Course on Meta-Heuristic Search Methods for Combinatorial Optimization Problems

Santosh Kumar Mandal, *Ph.D research fellow*

AutOrI LAB,  
DIA, Roma Tre  
Email: mandal@dia.uniroma3.it

January 20, 2014
Outline

1. Tabu search
2. Genetic algorithm
Tabu search

working process

---

Input: \(s^{(o)}\) - the initial solution;
Output: \(s^*\) - the best found solution;
Initialize the Tabu List \(T\);
set: Aspiration criteria;
set: \(s = s^{(o)}\) and \(s^* = s\);
Repeat
  Generate solutions in the neighborhood of \(s\);
  Select the best possible solution \(s' \notin T\) or satisfying the aspiration criteria;
  set \(s = s'\);
  Insert the solution \(s\) (or its attribute) into the tabu list \(T\);
if \(f(s) < f(s^*)\)
  set \(s^* = s\);
end if;
Update the tabu list \(T\);
Until (stopping condition is satisfied);

**Table:** Pseudocode of Tabu Search

**Figure:** Tabu search
Aspiration criteria:

- if the tabu solution is better than the best found solution.
- if the tabu solution possesses a particular attribute.
Tabu search components

Tabu list:
- A short term memory
- Stores visited solutions or moves/solutions attributes
- Prevents cycling
- The length of the list, called tabu tenure, controls diversification.
Types of Tabu list:

- **Static** [tabu tenure: 3-10].
- **Dynamic**: The size changes during the search in a given interval (Robust Tabu Search Algorithm)
- **Adaptive**: The size is increased or decreased according to the search information (e.g., Reactive Tabu search increases the tabu list if cycling occurs.)
An illustration on the travelling salesman problem (tabu tenure = 3):

Starting solution: Value = 234

```
1 2 3 4 5 6 7
2 5 7 3 4 6 1
```

Tabu list:

```
1 2 3 4 5 6 7
1 0 0 0 0 0 0 0
2 2 0 0 0 0 0 0
3 0 0 0 0 0 0 0
4 0 0 0 0 0 0 0
5 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0
```

**Figure:** Iteration 0
Current solution: Value = 234

2 5 7 3 4 6 1

After move: Value = 200

2 4 7 3 5 6 1

Tabu list:

Candidate list:

<table>
<thead>
<tr>
<th>Exchange</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4</td>
<td>-34</td>
</tr>
<tr>
<td>7.4</td>
<td>-4</td>
</tr>
<tr>
<td>3.6</td>
<td>-2</td>
</tr>
<tr>
<td>2.3</td>
<td>0</td>
</tr>
<tr>
<td>4.1</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure: Iteration 1
Tabu search

Tabu list

Current solution: Value = 200

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>7</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

Candidate list:

<table>
<thead>
<tr>
<th>Exchange</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>-2</td>
</tr>
<tr>
<td>2.3</td>
<td>-1</td>
</tr>
<tr>
<td>3.6</td>
<td>1</td>
</tr>
<tr>
<td>7.1</td>
<td>2</td>
</tr>
<tr>
<td>6.1</td>
<td>4</td>
</tr>
</tbody>
</table>

Choose move (3,1)

Tabu list:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure: Iteration 2
Tabu search

Current solution: Value = 200

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>3</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

Exchange | Value
--- | ---
3.1 | -2
2.3 | -1
3.6 | 1
7.1 | 2
6.1 | 4

Candidate list:

Choose move (3,1)

Update tabu list

Tabu list:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure: Iteration 2
**Tabu search**

**Tabu list**

Current solution: Value = 198

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Candidate list:

<table>
<thead>
<tr>
<th>Exchange</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3</td>
<td>2</td>
</tr>
<tr>
<td>2.4</td>
<td>4</td>
</tr>
<tr>
<td>7.6</td>
<td>6</td>
</tr>
<tr>
<td>4.5</td>
<td>7</td>
</tr>
<tr>
<td>5.3</td>
<td>9</td>
</tr>
</tbody>
</table>

Tabu list:

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure:** Iteration 3
Current solution: Value = 198

Tabu list:

Candidate list:

Choose move (2,4)

NB: Worsening move!

Update tabu list

Figure: Iteration 3
Current solution: Value = 202

Candidate list:

Exchange | Value
--- | ---
4.5 | -6
5.3 | -2
7.1 | 0
1.3 | 3
2.6 | 6

Tabu list:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>7</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

**Figure:** Iteration 4
Observations:

- In the example 3 out of 21 moves are prohibited.
- More restrictive tabu effect can be achieved by
  - Using stronger tabu-restrictions
  - Using OR instead of AND for the 2 cities in a move
Intensification:

- Use a medium term *recency memory*, which will memorize for each specified component the number of successive iterations the component is present in the visited solutions.
- Start the intensification process in a given period or after a certain number of iterations without improvement.
- Start the search with the best solution obtained, introducing the most visited component(s).
Diversification:

- Restart
- Continuous
- Strategic oscillation
Tabu search

diversification

Restart diversification:

- Use a long term frequency memory, which will memorize for each specified component the number of times the component is present in all visited solutions.
- Start the diversification process periodically or after a certain number of iterations without improvement.
- Start the search with the best solution obtained, introducing the least visited component(s).
Continuous diversification:

This is achieved by penalizing worsening moves.

\[ f(x) := f(x) + \delta_{\text{penalty}} \]  

(1.1)

For VRP (Taillard (1993)):

\[ \delta_{\text{penalty}} = \gamma \times \sqrt{mn} \times fr_u \]  

(1.2)

\[ fr_u: \text{ frequency of moving vertex } u \text{ in the past.} \]
Strategic oscillation:

- Proceed beyond the feasible boundary for a set depth.
- Turn around to enforce feasibility.

For CVRP:

\[ f(x) = f(x) + \alpha \times |Q(x)| \quad (1.3) \]

\( Q(x) \): total violation in the loading capacity.

**Figure:** strategic oscillation
Quick overview:

- Developed by Holland (1975).
- A population based meta-heuristic.
- Based on Darwinian’s principle of *competition*.
- A very successful algorithm, but not too fast.

*Figure*: GA illustration
Genetic algorithm

Components:

- Solution representation
- Population initialization
- Fitness function
- Parent selection mechanism
- Crossover and mutation operators
- Survivor selection
- Parameter settings
Solution representation:

In the Genetic algorithm, the encoded solution is referred as chromosome while the decision variables within a solution (chromosome) are genes. The possible values of variables (genes) are the alleles and the position of an element (gene) within a chromosome is named locus.

Table: Binary chromosome

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table: Permutation chromosome

| 10 | 8 | 7 | 1 | 2 | 5 | 3 | 9 | 4 | 6 |

Table: Real-valued chromosome

| 0.23 | 0.10 | 1.0  | 0.89 | 0.95 | 0.64 | 1.0  | 0.45 | 0.76 | 0.34 |
**Genetic algorithm components**

*Population initialization:*
- Uniformly at random
- Use any heuristic

**Facts:**
- Lower population size: poor solution quality
- Higher population size: good solution quality, but more computational time
- Find a satisfactory balance point.
Parent selection methods:

- Two widely used:
  - Tournament selection.
  - Roulette-wheel selection.
- Others:
  - Rank-based selection.
  - Sigma scaling
  - Boltzmann selection
**Roulette wheel procedure:**

**step 1:** Consider a roulette wheel and assign each solution \( i \) some portion of the wheel. The area of the wheel allocated to an individual solution \( i \) is equal to:

\[
\frac{\text{fitness}(i)}{\sum_i \text{fitness}(i)} \times 100 \% \quad (2.1)
\]

**step 2:** Rotate the wheel and select the solution corresponding to the selection point.

**step 3:** Inset a copy of the selected solution in the mating pool, and repeat the process until it is full (population size times).

**Figure:** Roulette wheel illustration
Genetic algorithm

selection methods

**Tournament selection:**

**step 1:** Draw $t$ solutions from the population and select the fittest one with some probability.

**step 2:** Insert a copy of the selected solution in the mating pool, and put solutions back into the population.

**step 3:** Repeat the process until the mating pool is full.
Tournament selection:

- **Deterministic tournament**: Always select the best one.
- **Binary tournament**: Only two players are involved.

Facts:

- Better in maintaining selection pressure than the Roulette wheel procedure.
- Tournament size should be set appropriately.
Genetic algorithm
components

Crossover/Recombination:

- It is the process of generating new solutions, called off-springs, by mixing genes of two or more parent solutions from the mating pool. The operation is executed with some probability.
- The crossover probability should be set high.
**Genetic algorithm**

**crossover**

**Figure:** An illustration of Order crossover [Oliver et al. (1987)]
**Figure**: An illustration of one-point crossover
Genetic algorithm crossover

\( \beta \): Spread factor

\( p_1 & p_2 \): Parent solutions

\( c_1 & c_2 \): Off-springs

\[ \beta = \left| \frac{c_1 - c_2}{p_1 - p_2} \right| \]

- **Contracting Crossover** \( \beta < 1 \)
  - The offspring points are enclosed by the parent points.

- **Expanding Crossover** \( \beta > 1 \)
  - The offspring points enclose the parent points.

- **Stationary Crossover** \( \beta = 1 \)
  - The offspring points are the same as parent points.
Simulated binary crossover [Agrawal and Deb (1994)]:

- High values of \( n \) will create off-springs near the parents; vice versa in the case of low values of \( n \).

\( n = 2 \) for mono - objective problems

\[
\begin{align*}
\text{contracting:} & \quad c(\beta) = 0.5(n + 1)\beta^n, \beta \leq 1 \\
\text{expanding:} & \quad c(\beta) = 0.5(n + 1)\frac{1}{\beta^{n+2}}, \beta > 1
\end{align*}
\]
Simulated binary crossover

Generates off-springs symmetrically about the parents.

\[ c_1 = \frac{p_1 + p_2}{2} - 0.5 \times \beta^* \times (p_2 - p_1) \]  \hspace{1cm} (2.2)

\[ c_2 = \frac{p_1 + p_2}{2} + 0.5 \times \beta^* \times (p_2 - p_1) \]  \hspace{1cm} (2.3)

To calculate \( \beta^* \):

- Generate a random number \( \mu \) in \([0, 1]\)
- Get \( \beta \) value that makes area under the curve = \( \mu \)
Genetic algorithm
components

Mutation:
In this process, the structure of off-spring generated via crossover is
further changed slightly. The mutation is also with some probability.

Facts:
- The mutation probability (i.e. probability of mutating each gene)
  should be set low.

\[ P_m = \frac{1}{L} \quad L: \text{length of string} \quad (2.4) \]

- Probability of mutating an individual becomes

\[ P_{\text{string}} = 1 - (1 - P_m)^L \quad (2.5) \]
Figure: An illustration of Inversion mutation
Figure: An illustration of flip mutation

| parent | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 |
| child  | 0 1 0 0 1 0 1 1 0 0 0 1 0 1 1 0 0 1 |
Genetic algorithm

Polynomial mutation:

\[ P(\delta): \text{Probability distribution function} \]
\[ \eta_m = 20 \text{ is generally used} \]

To calculate \( \delta_i \):
- Generate a random number \( \mu \) in \([0, 1]\)
- Get \( \delta \) value that makes area under the curve = \( \mu \)

\[ x'_i = x_i + (x^u_i - x^l_i) \delta_i \]

\[ P(\delta) = 0.5(\eta_m + 1)(1 - |\delta|^\eta_m) \]
Genetic algorithm components

Survivor selection:
» Select the top best individuals among parent and offspring solutions for the next generation of search.

Facts:
• This strategy promotes faster convergence.
• Premature convergence may occur.

