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 13, 2014
- **Combinatorial optimization problem**: A problem in which an optimal solution is sought among many possible alternative solutions.

- **Heuristic**: A method to discover solution of a problem.

- **Meta-heuristic**: A master strategy that enables a heuristic to adjust the balance between *diversification and intensification* for avoiding a local optimum.
- **Combinatorial optimization problem**: A problem in which an optimal solution is sought among many possible alternative solutions.

- **Heuristic**: A method to discover solution of a problem.

- **Meta-heuristic**: A master strategy that enables a heuristic to adjust the balance between *diversification and intensification* for avoiding a local optimum.
- **Combinatorial optimization problem**: A problem in which an optimal solution is sought among many possible alternative solutions.

- **Heuristic**: A method to discover solution of a problem.

- **Meta-heuristic**: A master strategy that enables a heuristic to adjust the balance between *diversification and intensification* for avoiding a local optimum.
• **Diversification**: Investigation of non-explored regions of the search space.
• **Intensification**: Exploitation of best found solutions to search thoroughly promising regions of the search space.
• **Local optimum**: An optimal solution within a neighbouring set of solutions.
• **Global optimum**: The optimal solution among all possible solutions.
- **Diversification**: Investigation of non-explored regions of the search space.
- **Intensification**: Exploitation of best found solutions to search thoroughly promising regions of the search space.
- **Local optimum**: An optimal solution within a neighbouring set of solutions.
- **Global optimum**: The optimal solution among all possible solutions.
• **Diversification**: Investigation of non-explored regions of the search space.

• **Intensification**: Exploitation of best found solutions to search thoroughly promising regions of the search space.

• **Local optimum**: An optimal solution within a neighbouring set of solutions.

• **Global optimum**: The optimal solution among all possible solutions.
- **Diversification**: Investigation of non-explored regions of the search space.
- **Intensification**: Exploitation of best found solutions to search thoroughly promising regions of the search space.
- **Local optimum**: An optimal solution within a neighbouring set of solutions.
- **Global optimum**: The optimal solution among all possible solutions.
It can solve large instances within a reasonable computational time.

- Meta-heuristics are general purpose algorithms.
- Advanced meta-heuristics use search experience.

**Figure:** optimization methods
Meta-heuristics are general purpose algorithms. Advanced meta-heuristics use search experience.

- It can solve large instances within a reasonable computational time.

Figure: optimization methods
Working process

| step 1: | Generate an initial solution \((s)\) |
| step 2: | Find \(s'(s' \in N(s))\) using a neighborhood operator \(\mu\) |
| step 3: | \(s \leftarrow s'\) (if \(s'\) is better) |
| step 4: | Repeat step 2 - step 3 until \((stopping criteria)\) |

**Table:** Template of basic local search
Neighbourhood

- More than one solution can be generated in the neighbourhood.
- Acceptance criteria:
  - First improvement
  - Best improvement

Figure: Cube-shaped neighbourhood

⇒ Neighbourhood operator is a systematic mechanism of changing the structure of a solution (e.g., Flip operator for binary strings).
Meta-heuristics are optimization methods that are designed to find solutions in complex and large solution spaces. They are particularly useful in situations where traditional methods fail due to the computational complexity of the problem. Here, we have a search space which includes:

- **Local optima**: These are the optimal solutions within a limited region of the search space.
- **Global optima**: These are the overall best solutions in the entire search space.
- **Initial solution**: The starting point of the search process.

The objective is to find the minimum of the function $f(X, Y)$. The diagram illustrates the search process moving from an initial solution towards the global optima.
Overview

It is a heat treatment process, whereby a metal is heated to a specific temperature and then allowed to cool slowly. The process occurs by the diffusion of atoms and produces a minimum energy crystalline structure.

Figure: Microscopic structure of steel
Developed by Kirkpatrick et al. (1983)

<table>
<thead>
<tr>
<th>Thermodynamics</th>
<th>Simulated annealing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crystal structure</td>
<td>Solution</td>
</tr>
<tr>
<td>Energy</td>
<td>Fitness function</td>
</tr>
<tr>
<td>Structure change</td>
<td>Moving to a new solution</td>
</tr>
<tr>
<td>Temperature</td>
<td>Control parameter</td>
</tr>
<tr>
<td>Equilibrium structure</td>
<td>Best solution</td>
</tr>
</tbody>
</table>

Table: Metaphor
Template of simulated annealing algorithm.

**Input:** Cooling schedule.

\[
s = s_0 ; /* \text{Generation of the initial solution} */
\]

\[
T = T_{\text{max}} ; /* \text{Starting temperature} */
\]

**Repeat**

**Repeat** /* At a fixed temperature */

Generate a random neighbor \( s' \);

\[
\Delta E = f(s') - f(s);
\]

**If** \( \Delta E \leq 0 \) **Then** \( s = s' \) /* Accept the neighbor solution */

**Else** Accept \( s' \) with a probability \( e^{\frac{-\Delta E}{T}} \);

**Until** Equilibrium condition

/* e.g. a given number of iterations executed at each temperature \( T \) */

\[
T = g(T) ; /* \text{Temperature update} */
\]

**Until** Stopping criteria satisfied /* e.g. \( T < T_{\text{min}} \) */

**Output:** Best solution found.

---

**Figure:** Pseudocode of Simulated annealing
Temperature effect

Slower is the cooling, better will be the quality of the final solution.

Figure: Effect of temperature
Basic Meta-heuristic
Local search
Simulated annealing algorithm
Assignment
Simulated annealing algorithm
References

Physical annealing process
Overview
## Cooling schedule

- **Linear**

  \[ T = T - \beta \quad \beta; \text{a constant value} \quad (4.1) \]

- **Geometric**

  \[ T = T \times \alpha \quad \alpha \in [0.5, 0.99] \quad (4.2) \]

- **Logarithmic**

  \[ T_i = \frac{T_{\text{initial}}}{\log(i)} \quad i: \text{iteration number(outerloop)} \quad (4.3) \]
very slow

\[ T_{i+1} = \frac{T_i}{1 + (\beta \times T_i)} \]  \hspace{1cm} (4.4)

\[ \beta = T - \frac{T_{\text{final}}}{(L - 1) \times T_{\text{initial}} \times T_{\text{final}}} \]  \hspace{1cm} (4.5)

\( L \): Number of transitions in the inner loop. It should be set according to the size of the problem/neighbourhood size.
Key issues

- Solution representation.
- Initial solution.
- A neighbourhood operator (should generate a valid solution).
- Cooling schedule.
- Length of the inner loop.
Assignment-I
Threshold accepting [Dueck and Scheuer (1990)]:

- Acceptance probability

\[ P(s', s) = \begin{cases} 
1 & \text{if } \Delta E \leq Q_{value} \\
0 & \text{otherwise} 
\end{cases} \]

- Update \( Q_{value} \) according to an annealing schedule.

- Faster than Simulated annealing
Record-to-record

Record-to-record travel algorithm:

*RECORD*: objective value of the best found solution.

---

Template of the record-to-record travel algorithm.

**Input:** Deviation $D > 0$.

$s = s_0$; /* Generation of the initial solution */

$RECORD = f(s)$; /* Starting RECORD */

**Repeat**

Generate a random neighbor $s'$;

*If* $f(s') < RECORD + D$ *Then* $s = s'$; /* Accept the neighbor solution */

*If* $RECORD > f(s')$ *Then* $RECORD = f(s')$; /* RECORD update */

Until Stopping criteria satisfied

**Output:** Best solution found.
Great deluge algorithm [Dueck (1993)]:

- The climber will try to reach at the top (global optima position).
- The climber will try to keep his/her foot above the water level.
**Table:** Pseudocode of Great deluge algorithm

---

**Input:** Water Level;

\[ s = s_0 \]

Generation of the initial solution;

Choose the rain speed UP;

Choose the initial water level;

**Repeat**

- Generate a neighbour solution \( s' \);
  
  if \( f(s') > Level \), then \( s = s' \);

- Level = Level + UP;

**Until** (stopping criteria)

Output: Best found solution
