(Meta)heuristics 1

Cooking Receipes or Experimental Sciences

Van-Dat CUNG

Heuristics

- Heuristic = **approximate method**
- Method to provide a **good quality** of solution, but **not necessary optimal**
- Even if the solution is optimal, the method **does not provide any proof of its optimality (epsilon-approximation excepted)**
- Heuristic methods are generally **specialized** in solving a specific problem such as Clarke and Wright’s algorithm for CVRP

Metaheuristics

- Approximate methods, but can be applied to many problems:

  General and generic approximate solving concept
- **Many metaheuristics** have been proposed since the 80's
- **The most well-knowns**:
  - Simulated Annealing (Statistical Physics)
  - Tabu Search (Artificial Intelligence)
  - Genetic Algorithms (Bioinformatics)
  - Variable Neighborhood Search (CO)
  - Ant Colonies/Systems (Bioinformatics)

Families of Metaheuristics

- Constructive methods (Greedy, Ant Colonies)
- Local Searches (Hill Climbing, Simulated Annealing, Tabu Search)
- Evolutionary/Population based (Genetic Algorithms, Ant Colonies, System Multi-Agent)
- Hybrids (Grasp, Scatter Search, etc.)

When should they be employed ?

- On (NP-)hard problems
- Good solutions are wanted (not the proof of optimality)
- Short computing times
- Constraints and/or objective functions may not be linear (no viable solution in LP or MILP)
- But caution… in result analyses

Greedy methods

- **Constructive** methods
- **No initial solution**
- Can be viewed as a **depth-first search without backtracking**
- At each iteration or constructive step, the **best choice** is made to get the **best partial solution** (set of solutions reducing iteration after iteration to only a single solution)
- E.g. the nearest neighbor in TSP

The nearest neighbor in TSP

- Initialize the total length of the tour lg=0; k=0;
- Start from city i at random;
- While k < #city Do
  - Look for the city j nearest to city i without subtour;
  - lg=lg+d(i,j);
  - i=j; // iterate from the city j
  - k=k+1;
- End While

Ant Colony [Dorigo, 1992]

- **Population based constructive method with memory** (in sequential → iterated greedy)
- An **individual** = an **ant**
- Each ant **builds a complete solution**
- The **population** = the **colony**
- The **memory** = **pheromone trails**

Ant Colony for TSP

- Initialization (distance matrix)
- For *t*=1 to *T* Do // arbitrary stop criterion
  - For *k*=1 to *m* Do // m ants
    - Repeat until ant *k* builds a complete tour
      - Choice for the next nearest city *j* with proba. p_{ij} (eq. [1])
    - Let L_k be the length of the tour generated by ant *k*
  - End For
  - Update pheromones τ_{ij} on all edges (eq. [2])
- End For

Eq. [1] to compute p_{ij}

- τ_{ij} intensity of pheromones between cities *i* and *j*
- α regulation parameter of the influence of τ_{ij}
- $\eta_{ij} = \frac{1}{d_{ij}}$ visibility of city *j* from city *i*
- β regulation parameter of the influence of η_{ij}
- Ω set of remaining cities to visitr

$$p_{ij} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{h \in \Omega}[\tau_{ih}]^{\alpha}[\eta_{ih}]^{\beta}} \quad j \in \Omega \quad \text{otherwise} \;\; p_{ij} = 0$$

Eq. [2] to compute τ_{ij}

- t iteration counter
- $\rho \in [0,1]$ evaporation regulation parameter of τ_{ij}
- $\Delta\tau_{ij}$ total pheromone change on edge (i,j)
- m number of ants
- $\Delta\tau_{ij}^{k}$ pheromone change on edge (i,j) by ant k
- L_k length of the tour found by ant k

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \qquad \tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}$$

$\Delta\tau_{ij}^{k} = \dfrac{1}{L_k}$ if ant k uses edge (i,j), 0 otherwise.