Co-funded by the
Erasmus+ Programme
of the European Union

Advanced Optimization: Techniques and Industrial Applications

Module 2: Heuristics and Metaheuristics

Curriculum Development
of Master's Degree Program in
Industrial Engineering for Thailand Sustainable Smart Industry

MSIE 4.0

Co-funded by the
Erasmus+ Programme
of the European Union

Session 2.3:

Local Search Methods: ALNS and Tabu Search

Universidade do Minho

Curriculum Development
of Master's Degree Program in
Industrial Engineering for Thailand Sustainable Smart Industry

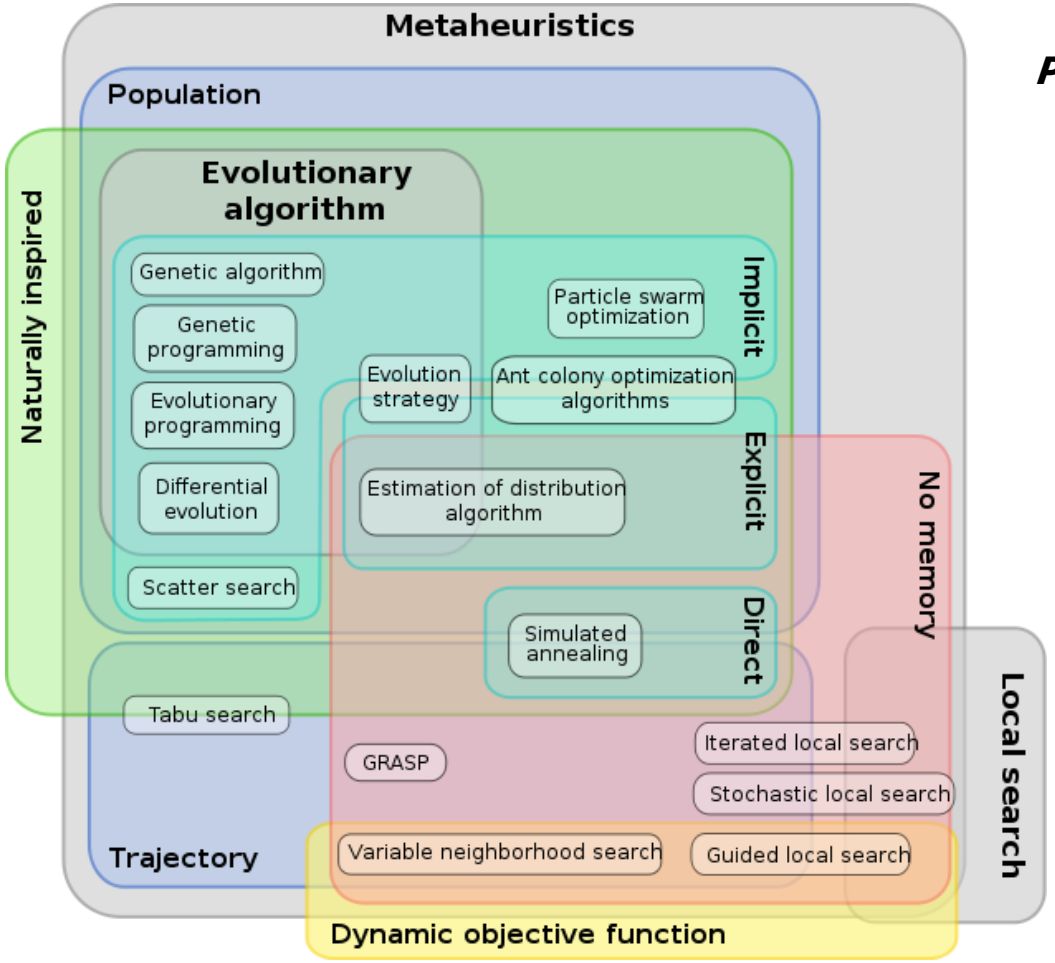
Heuristics and Mataheuristics

- **Metaheuristics**: High-level strategies for guiding a search (Glover, 1986) according to feedback from the objective function, previous decisions and prior performance (Stutzle, 1998). In other words, the rules of a meta-heuristic consist of (at least) two orders, such that **the overall searching behavior keeps changing** while exploring the state-space.

Examples: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Adaptive Large Neighborhood Search (ALNS)

Co-funded by the
Erasmus+ Programme
of the European Union

Metaheuristics Classification



Par nojhan le vendredi, 2007

Par nojhan le vendredi, 2007

Co-funded by the Erasmus+ Programme of the European Union

Large Neighborhood Search (LNS)

- Proposed by Shaw (1998)
- LNS is a concept for efficient search developing from local search by improving searching techniques within a small to large area
- The principles of LNS are based on two main processes: destroy and repair methods.
- A destroy method destructs part of the current solution while a repair method rebuilds the destroyed solution

Co-funded by the
Erasmus+ Programme
of the European Union

Large Neighborhood Search (LNS)

Algorithm 1 Large neighborhood search

```
1:  input: a feasible solution x
2:  x^b = x;
3:  repeat
4:      x^t = r(d(x));
5:      if accept(x^t, x) then
6:          x = x^t;
7:      end if
8:      if c(x^t) < c(x^b) then
9:          x^b = x^t;
10:     end if
11: until stop criterion is met
12: return x^b
```

x : current feasible solution

x^b : The best solution of the search

x^t : a temporary solution

function $d(\ )$: destroy method

function $r(\ )$: repair method

Co-funded by the Erasmus+ Programme of the European Union

Adaptive Large Neighborhood Search (ALNS)

- Proposed by S. Ropke and D. Pisinger (2006)
- Extend LNS by allowing multiple destroy and repair methods to be used within the same search
- Each destroy/repair method is assigned a weight that controls how often the particular method is attempted during the search
- The weights are adjusted dynamically as the search progresses so that the heuristic adapts to the instance at hand and to the state of the search

Co-funded by the
Erasmus+ Programme
of the European Union

Adaptive Large Neighborhood Search (ALNS)

Algorithm 2 Adaptive large neighborhood search

```
1:  input: a feasible solution x
2:  x^b = x; ρ^- = (1,...,1); ρ^+ = (1,...,1);
3:  repeat
4:      select destroy and repair methods d ∈ Ω^- and r ∈ Ω^+ using ρ^- and ρ^+;
5:      x^t = r(d(x));
6:      if accept(x^t, x) then
7:          x = x^t;
8:      end if
9:      if c(x^t) < c(x^b) then
10:         x^b = x^t;
11:     end if
12:     update ρ^- and ρ^+;
13: until stop criterion is met
14: return x^b
```

x : current feasible solution

x^b : The best solution of the search

Ω^- : set of destroy method

Ω^+ : set of repair method

x^t : a temporary solution

function $d( )$: destroy method

function $r( )$: repair method

ρ^- and ρ^+ restore the weight of each destroy and repair method, respectively

Co-funded by the Erasmus+ Programme of the European Union

Adaptive Large Neighborhood Search (ALNS)

What extend from LNS…

- In line 2: ρ^- and ρ^+ is introduced to restore the weight of each destroy and repair method, respectively. Initially all methods have the same weight.
- In line 4, the weight vectors, ρ^- and ρ^+ are used to select the destroy and repair methods using a ***roulette wheel principle***.

Co-funded by the
Erasmus+ Programme
of the European Union

Adaptive Large Neighborhood Search (ALNS)

- The algorithm calculates the probability Φ_j^- of choosing the j^{th} destroy method as follows

$$\phi_j^- = \frac{\rho_j^-}{\sum_{k=1}^{|\Omega^-|} \rho_k^-}$$

and the probabilities for choosing the repair methods are determined in the same way

Co-funded by the
Erasmus+ Programme
of the European Union

Adaptive Large Neighborhood Search (ALNS)

- The weights are adjusted dynamically, based on the recorded performance of each destroy and repair method. This takes place in line 12: when an iteration of the ALNS heuristic is completed, a score y for the destroy and repair method used in the iteration is computed using the formula

$$\psi = \max \begin{cases} \omega_1 & \text{if the new solution is a new global best,} \\ \omega_2 & \text{if the new solution is better than the current one,} \\ \omega_3 & \text{if the new solution is accepted,} \\ \omega_4 & \text{if the new solution is rejected,} \end{cases} \quad (1)$$

Where $\omega_1, \omega_2, \omega_3,$ and ω_1 are parameters. A highly value corresponds to a successful method.

Co-funded by the Erasmus+ Programme of the European Union

Adaptive Large Neighborhood Search (ALNS)

- Let *a* and *b* be the indices of the destroy and repair methods that were used in the last iteration of the algorithm, respectively. The components corresponding to the selected destroy and repair methods in the r− and r+ vectors are updated using the equations

$$\rho_a^- = \lambda \rho_a^- + (1-\lambda)\psi, \quad \rho_b^+ = \lambda \rho_b^+ + (1-\lambda)\psi, \qquad (2)$$

- where λ ϵ [0,1] is the *decay* parameter that controls how sensitive the weights are to changes in the performance of the destroy and repair methods.
- Note that the weights that are not used at the current iteration remain unchanged.
- The aim of the adaptive weight adjustment is to select weights that work well for the instance being solved.

Co-funded by the
Erasmus+ Programme
of the European Union

Example Applications of LNS and ALNS

- Routing problems
  - TSP (Traveling salesman problem)
  - VRP (Vehicle routing problem)
  - VRPTW (Vehicle routing problem with time windows)
  - PDPTW (Pickup and delivery problem with time windows)
- Scheduling problems
  - Single machine scheduling
  - Cumulative scheduling
  - Job shop scheduling
  - Constrained project scheduling problem

Co-funded by the
Erasmus+ Programme
of the European Union

Tabu Search

- A chief way to exploit memory in tabu search is to classify a subset of the moves in a neighborhood as forbidden (or ***tabu***).
- A ***neighborhood*** is constructed to identify adjacent solutions that can be reached from current solution.
- The classification depends on the history of the search, and particularly on the recency or frequency that certain move or solution components, called ***attributes***, have participated in generating past solutions.
- A ***tabu list*** records forbidden moves, which are referred to as ***tabu moves***.
- Tabu restrictions are subject to an important exception. When a tabu move has a sufficiently attractive evaluation where it would result in a solution better than any visited so far, then its tabu classification may be overridden. A condition that allows such an override to occur is called an ***aspiration criterion***.

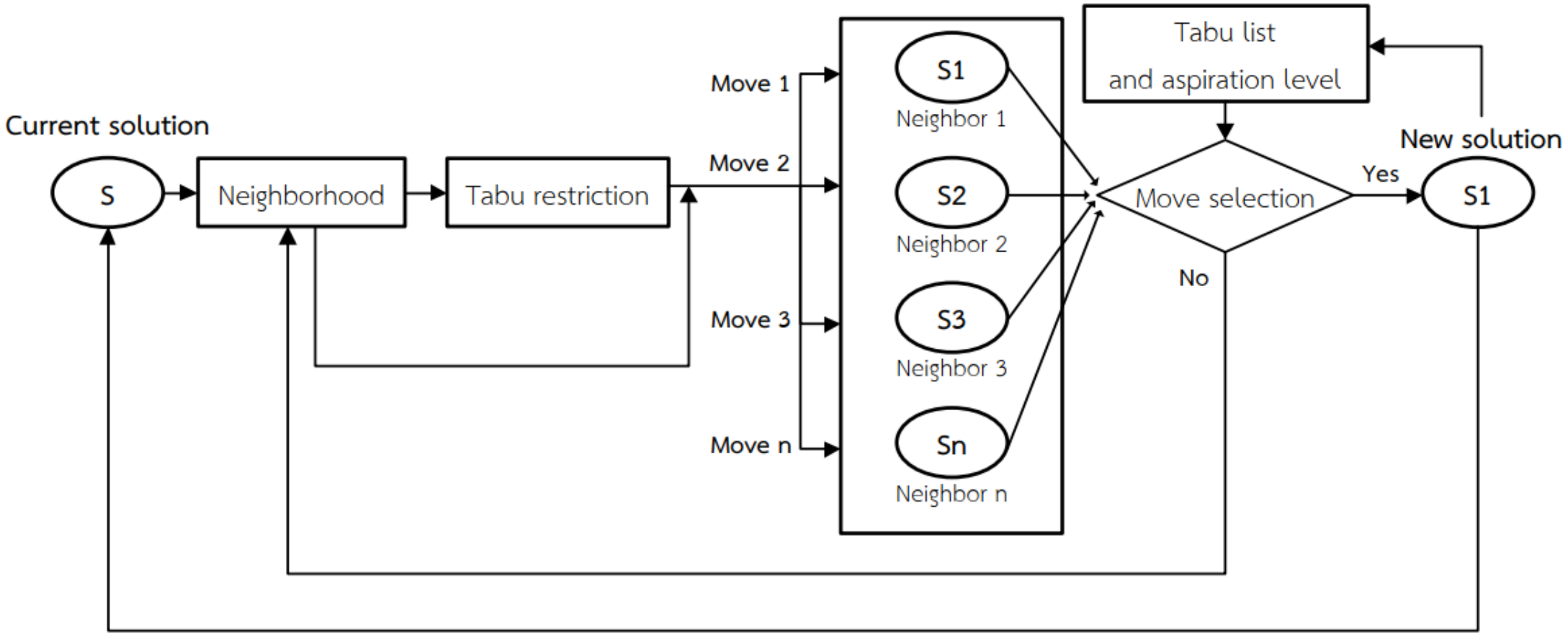
Co-funded by the
Erasmus+ Programme
of the European Union

Basic Tabu Search Algorithm

- Step 1: Choose an initial solution i in S. Set i* = i and k=0.
- Step 2: Set k=k+1 and generate a subset V* of solution in N(i,k) such that either one of the Tabu conditions is violated or at least one of the aspiration conditions holds.
- Step 3: Choose a best j in V* and set i=j.
- Step 4: If f(i) < f(i*) then set i* = i.
- Step 5: Update Tabu and aspiration conditions.
- Step 6: If a stopping condition is met then stop. Else go to Step 2.

Co-funded by the
Erasmus+ Programme
of the European Union

Tabu Search Algorithm (TS)

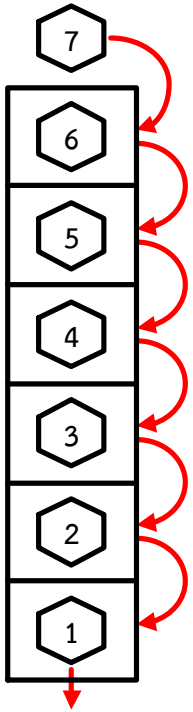


Co-funded by the
Erasmus+ Programme
of the European Union

Tabu Search Algorithm (TS)

Tabu List: T

First in First out: FIFO



Stopping criteria

- Maximum number of iterations
- Maximum computation time
- Number of consecutive iterations without an improvement
- Objective function setting

Co-funded by the
Erasmus+ Programme
of the European Union

Example of TS

- Asymmetric traveling salesman problem: ATSP. Find the routing of this problem by TS. Tabu list = 2 , Neighborhood size = 4
- Iteration = 3 and Swap move

From/To	City 1	City 2	City 3	City 4	City 5
City 1	0	4	7	9	12
City 2	3	0	8	12	15
City 3	7	9	0	17	12
City 4	8	12	13	0	16
City 5	12	6	15	18	0

Co-funded by the
Erasmus+ Programme
of the European Union