PA184 - Heuristic Search Methods

Lecture 4 – Meta-heuristics

- Motivation for Meta-heuristics
- Single-solution Meta-heuristics
  - Iterated Local Search
  - Threshold Acceptance
  - Great Deluge
  - Simulated Annealing

Learning outcomes:

- Understand the purpose of meta-heuristics methods
- Understand the main classification of meta-heuristics
- Compare different strategies for accepting non-improving solutions in meta-heuristic search
- Analyse the search strategy of some popular meta-heuristics

Motivation for Meta-heuristics

Greedy heuristics and simple local search have some limitations with respect to the quality of the solution that they can produce.

A good balance between intensification and diversification is crucial in order to obtain high-quality solutions for difficult problems.

A meta-heuristic can be defined as "an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method"(Voss et al. 1999).

A meta-heuristic method is meant to provide a generalised approach that can be applied to different problems. Meta-heuristics do not need a formal mathematical model of the problem to solve.

Reasons for Using Meta-heuristic Methods

Using meta-heuristics is not justified if:

- An efficient algorithm exists for the problem
- The problem size is small enough that even a non-efficient algorithm can produce results in practical time.

The computational difficulty for solving an optimisation problem is determined by:

- The problem complexity class.
- The size of the problem instances being tackled.
- The particular structure of the problem instances being tackled.
- The available computation time to produce a solution.

Powerful commercial and free solvers exist nowadays. Examples include: cxplex, lingo, gurobi, xpress, lp solve, etc. However, for the same problem type, some instances might be too difficult for such solvers and meta-heuristics are a good alternative.

Types of Meta-heuristics

Single-solution method vs. Population-based methods

Nature-inspired method vs. Non-nature inspired methods

'Pure' methods vs. Hybrid methods

Memory-based vs. Memory-less methods

Deterministic vs. Stochastic methods

Iterative vs. Greedy methods

Examples of Meta-heuristics

The following algorithms are examples of meta-heuristics:

- Iterated local search
- Threshold acceptance
- Great deluge
- Simulated annealing
- Greedy randomised search procedure
- Guided local search
- Variable neighbourhood search
- Tabu search
- Evolutionary algorithms
- Particle swarm optimisation
- Artificial immune systems
- Etc.

Single-solution Meta-heuristics

These methods maintain one current solution at a time and conduct the search moving from one solution to another while building a single-point trajectory.

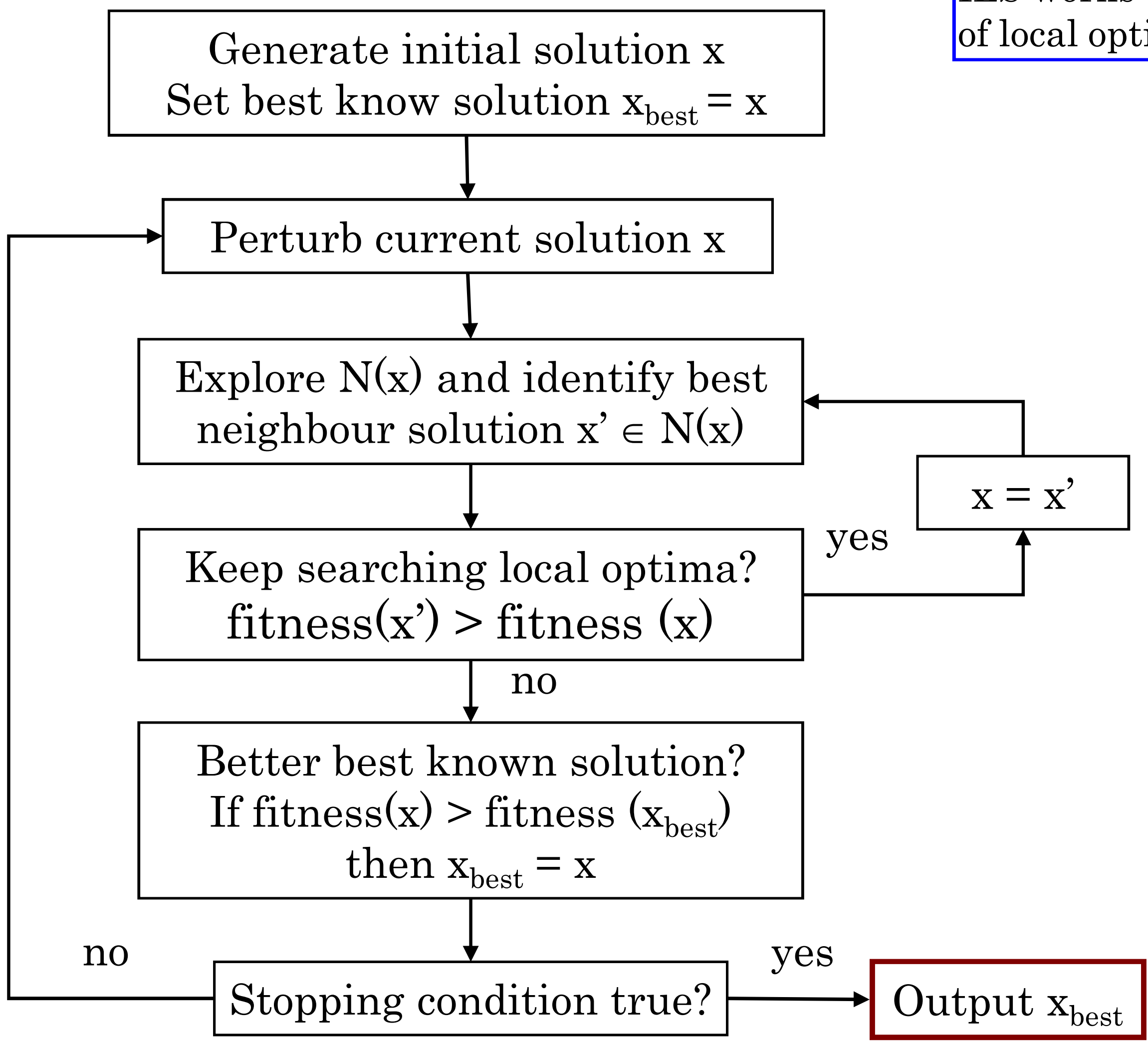
Important common components in single-solution meta-heuristics:

- Generation of candidate solutions (random, greedy, peckish, etc.)
- Definition of neighbourhoods (exchange, swaps, insertion, inversion, very large, ejection chains, cyclic, etc.) with strong *locality*
- Computation of objective function (exact, delta, approximate or surrogate, etc.)
- Replacement of current solution (deterministic, probabilistic, only improving, non-improving, etc.)
- Termination criteria (execution time, total iterations, total function evaluations, idle iterations, etc.)
- Parameter tuning (temperature, water level, memory length, etc.)

Designing good components is perhaps more important than tuning parameters

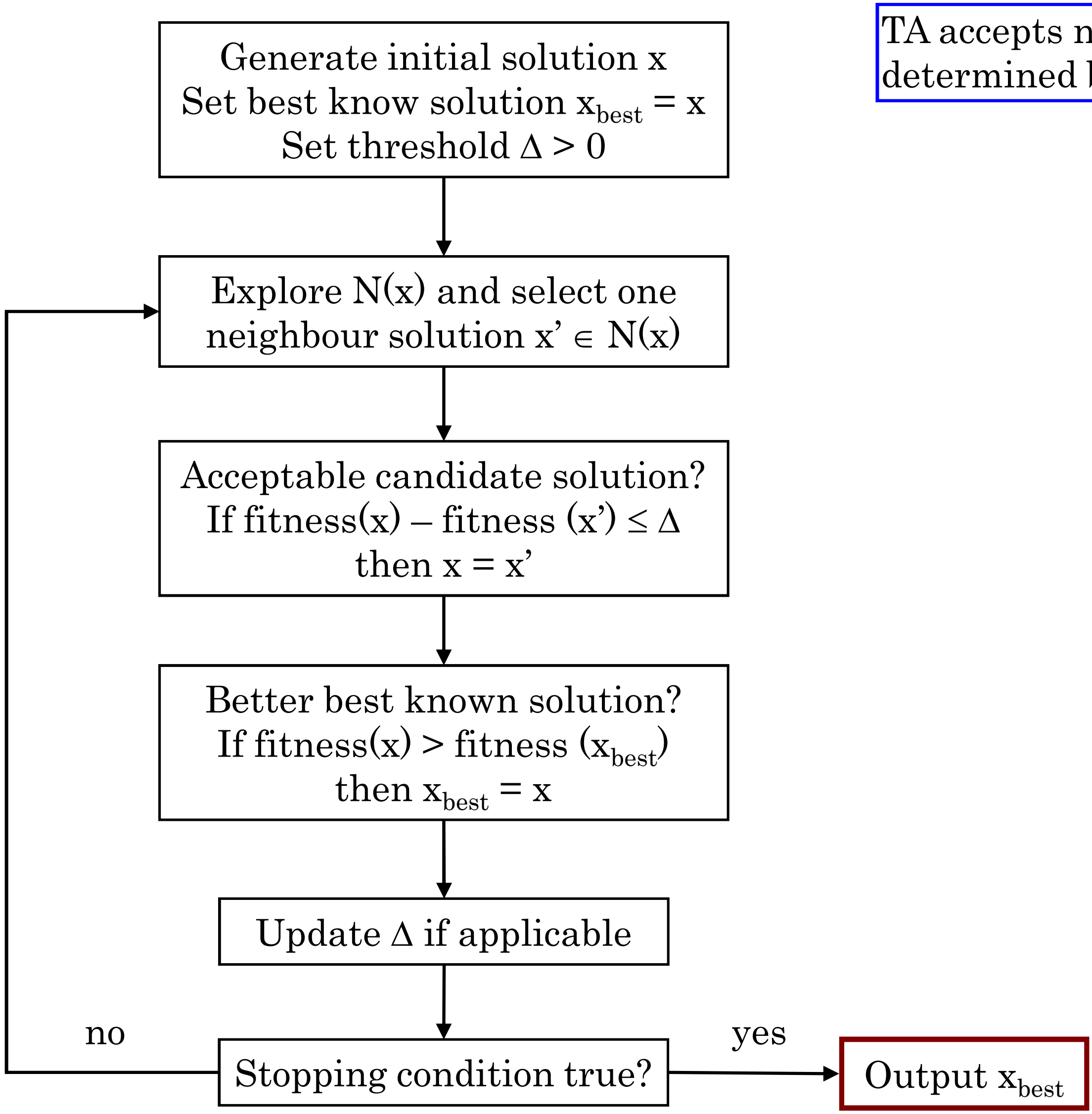
Iterated Local Search

ILS works on perturbations of local optima



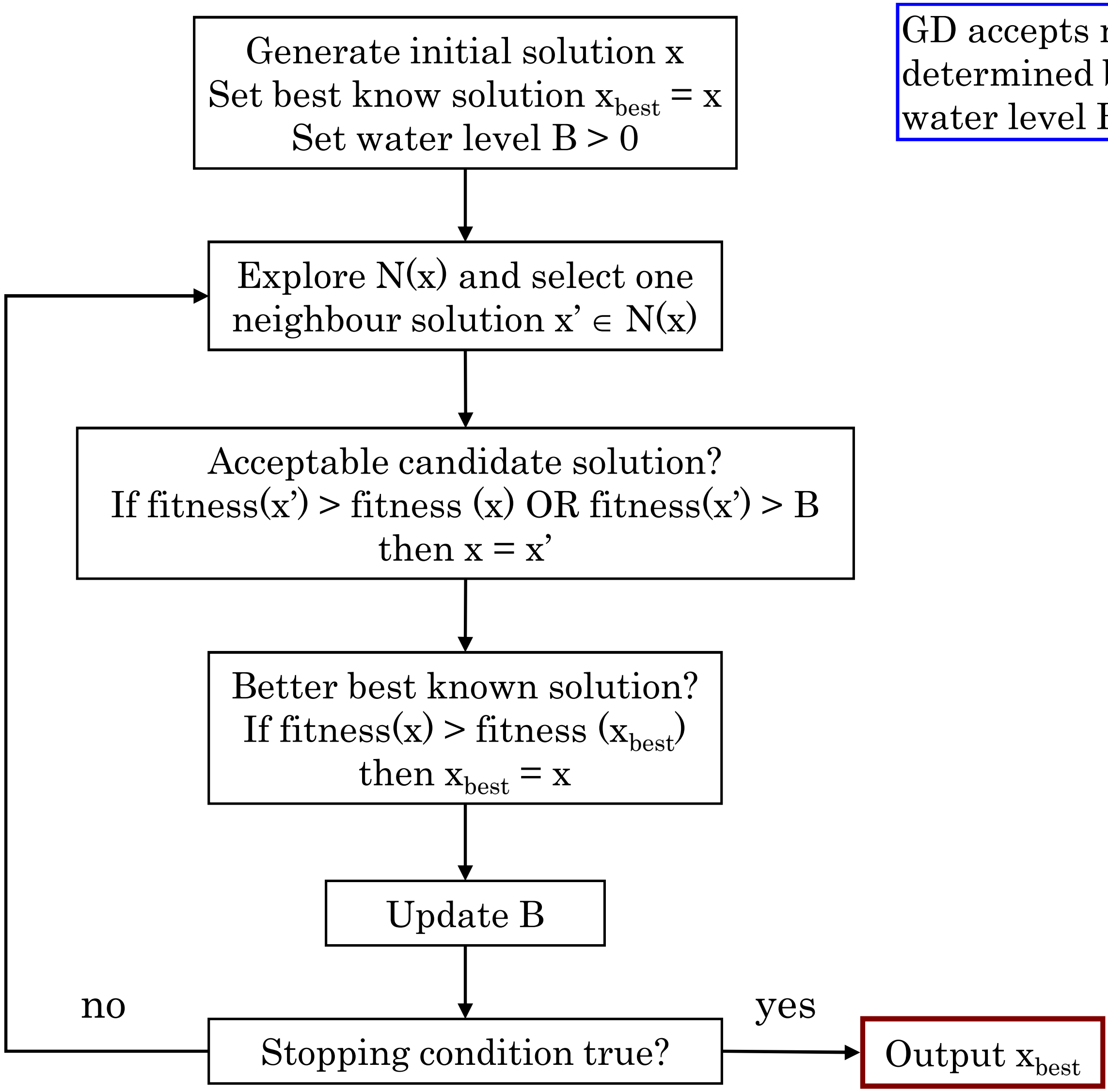
Threshold Acceptance

TA accepts non-improving as determined by a threshold



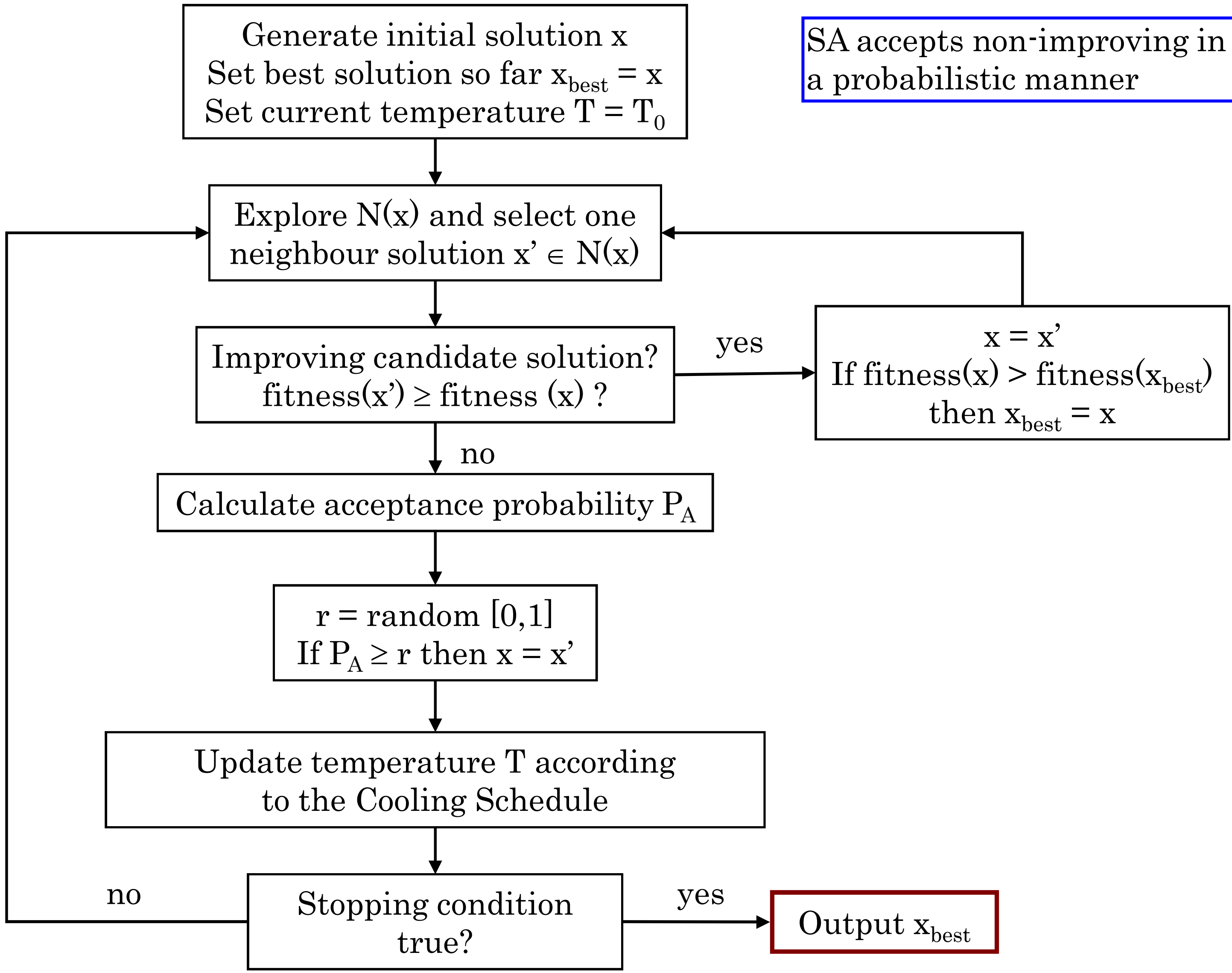
Great Deluge

GD accepts non-improving as determined by a changing water level B



Simulated Annealing

SA accepts non-improving in a probabilistic manner



Cooling Schedule in SA

In general, the current temperature T_i is determined by:

- Initial temperature T_0
- Final temperature T_F
- Decrement step t_{step}, i.e. number of iterations between temperature decrements.
- Cooling factor ΔT, i.e. the proportion of the temperature reduction.
- Re-heating step t_{reheat}, i.e. number of iterations after which the temperature is increased to the initial temperature or to another value.

The arithmetic cooling schedule modifies the temperature as follows.

$$\text{every } t_{step} \text{ iterations:} \quad T_i = T_{i-1} - \Delta T$$

$$\text{after } t_{reheat} \text{ iterations:} \quad T_i = T_0$$

the initial temperature is set to a high value in the standard SA algorithm, but it can also be set to a low value or to zero.

The geometric cooling schedule modifies the temperature as follows.

$$\text{every } t_{step} \text{ iterations}: \quad T_i = \alpha T_{i-1} \quad \text{where } 0 < \alpha < 1$$

alternatively

$$\text{every } t_{step} \text{ iterations}: \quad T_i = \frac{\alpha T_{i-1}}{1 + \alpha T_{i-1}} \quad \text{where } 0 < \alpha < 1$$

Note: re-heating can also be performed

The quadratic cooling schedule modifies the temperature as follows.

$$\text{every } t_{step} \text{ iterations}: \quad T_i = ai2 + bi + c \quad \text{where } a = \frac{T_0 T_F}{I_{total}}, b = \frac{T_F - T_0}{I_{total}}, c = T_0$$

Typically, the acceptance probability is calculated as follows.

$$P_A = \exp^{-\left(\Delta fitness / T_i\right)}$$

Additional Reading

Chapter 2 of the book (Talbi, 2009)

Corresponding chapters of the books (Glover and Kochenberger,2003) and (Burke and Kendall,2005)

G. Dueck. *New optimization heuristics: the great deluge algorithm and the record-to-record travel*. Journal of computational physics, Academic press, 104, 86-92, 1993.

G. Dueck, T. Scheuer. *Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing*. Journal of computational physics, Academic press, 90, 161-175, 1990.

F. Glover, E. Taillard, D. De Werra. *A user's guide to tabu search*. Annals of operations research, 41, 3-28, 1993.

Seminar Activity 4

The purpose of this seminar activity is to outline the design of a Tabu Search method for the GAP problem of Lecture 1.

Do the following:

1. Read the following article:

Juan A. Diaz, Elena Fernandez. A tabu search heuristic for the generalized assignment problem. *European Journal of Operational Research*, Vol. 132, pp. 22-38, 2001.

1. Outline a flow diagram of the Tabu Search method.

2. Describe the way in which the following components of this method were implemented:
   a) Tabu List
   b) Aspiration criterion
   c) Short-term and long-term memory
   d) Any other important issue