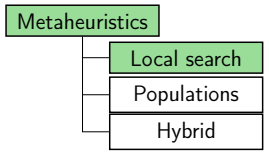
Multi-Neighborhood Metaheuristics



Multi-neighborhood metaheuristics are based on local search

A **multi-neighborhood** is obtained by the union of atomic local search neighborhoods

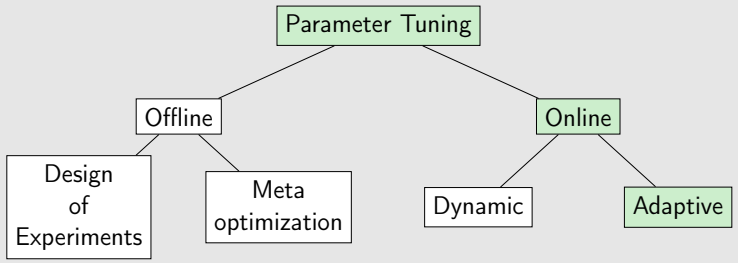
$$\mathcal{N} = \bigcup_k \mathcal{N}_k$$

In **stochastic** search, neighborhoods $\mathcal{N}_k$ are assigned with probabilities $p_k \in [0, 1]$, such that $\sum_k p_k = 1$

Probabilities $\{p_1 \dots p_n\}$ are key parameters with strong implications on the search trajectory of the metaheuristic.

Parameter Tuning

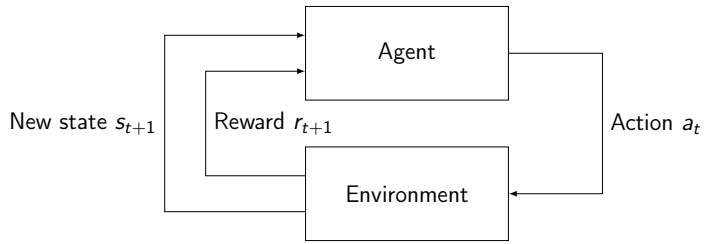
Parameter Tuning: procedure for the determination of (sub-)optimal values for the metaheuristic parameters



In adaptive **online learning**, parameters are adapted during the search itself, according to the search progress.

Reinforcement learning

Reinforcement learning: an intelligent agent that interacts with the environment learns the best policy for its actions to maximize the **cumulative reward**.



Q-Learning

Q-Learning: a reinforcement learning framework for **online tuning** of multi-neighborhood probabilities

$$p_{i,t} = \underbrace{(1-\alpha) \cdot p_{i,t-1}}_{\text{memory of past actions}} + \underbrace{\alpha \cdot r_{i,t}}_{\text{learning from last actions}}$$

- $p_{i,t}$: probability of neighborhood i at iteration t
- α : learning rate, $\alpha \in [0, 1]$
- $r_{i,t}$: reward obtained by neighborhood i at iteration t

Critical design choices

- *Reward function* $r_{i,t}$
- *Learning batch*: every how many iterations the agent learns?
- *Exploration/exploitation* trade-off:
  - *ϵ-greedy*: neighborhoods are chosen uniformly with probability ϵ, while with probability $1 - \epsilon$ the choice is done according to the policy.
  - *Minimum probability threshold*: $p_{i,t} = \max((1-\alpha) \cdot p_{i,t-1} + \alpha \cdot r_{i,t}, \tau)$

In Multi-Neighborhood Simulated Annealing, probabilities can be adapted through the Q-learning strategy

procedure *SimulatedAnnealing*(SearchSpace $\mathcal{S}$, **Multi-Neighborhood** $\mathcal{N} = \bigcup_{k=1}^{n} \mathcal{N}_k$, CostFunction F, Parameters T_0, T_f, α, N_s, $\{p_1 \dots p_n\}$)

$T \leftarrow T_0$
$s \leftarrow RandomState(\mathcal{S})$
$s_{best} \leftarrow s$
while $T \geq T_f$
  $n_s \leftarrow 0$
  while $n_s < N_s$
    $\mathcal{N}_k \leftarrow RandomNH(\mathcal{N}, \{p_1 \dots p_n\})$, $m \leftarrow RandomMove(s, \mathcal{N}_k)$
    $\Delta F \leftarrow F(s \oplus m) - F(s)$
    if $(\Delta F \leq 0)$
      $s \leftarrow s \oplus m$
      if $(F(s) < F(s_{best}))$
        $s_{best} \leftarrow s$
    else
      if $(RandomReal(0, 1) < e^{-\Delta F / T})$
        $s \leftarrow s \oplus m$
    $n_s \leftarrow n_s + 1$
  $T \leftarrow T \cdot \alpha$
return s_{best}

Currently on-going work

Multi-Neighborhood Metaheuristics

- Q-Learning Multi-Neighborhood **Simulated Annealing** for
  - Home Healthcare Routing and Scheduling problem
  - Sports Timetabling ITC2021 problem
  - Minimum Interference Frequency Assignment problem

Other methods

- Q-Learning on a multi-greedy **Construct, Merge, Solve and Adapt** for the Maximum Disjoint Dominating Sets Problem
  - Statistical tests confirmed that multi-greedy CMSA with Q-Learning improves over traditional CMSA, for certain compositions of the multi-greedy