universität innsbruck

Advanced Distributed Systems

Lecture 06-Optimization methods

Dr. Zahra Najafabadi Samani

Agenda

- Optimization as a search
- Problem characteristics
- Exact optimization approaches
- Heuristic approaches
- Metaheuristic approaches
- Selecting the optimizer

Optimization as a search

Problem components

Optimization design decisions typically involve making choices to maximize or minimize certain objectives while satisfying specific constraints.

1. Assign *m* continuous and *n* binary **design decisions**:

$$D_c \in Q^m, D_b \in \{0,1\}^n$$

2. Subject to a set of **constraints**

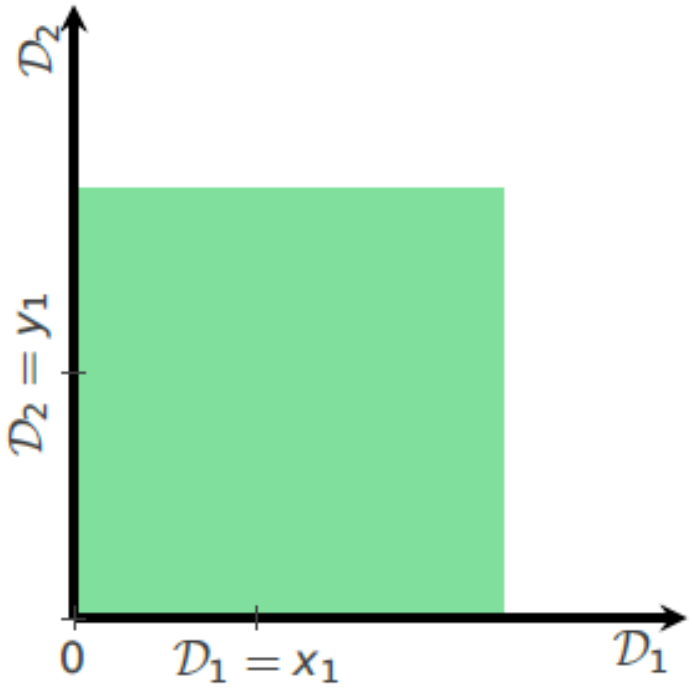
$$a_c \,.\, a_b \leq C$$

3. So that a set of **objective functions** minimized or maximized.

$$F(D_c, D_b)$$

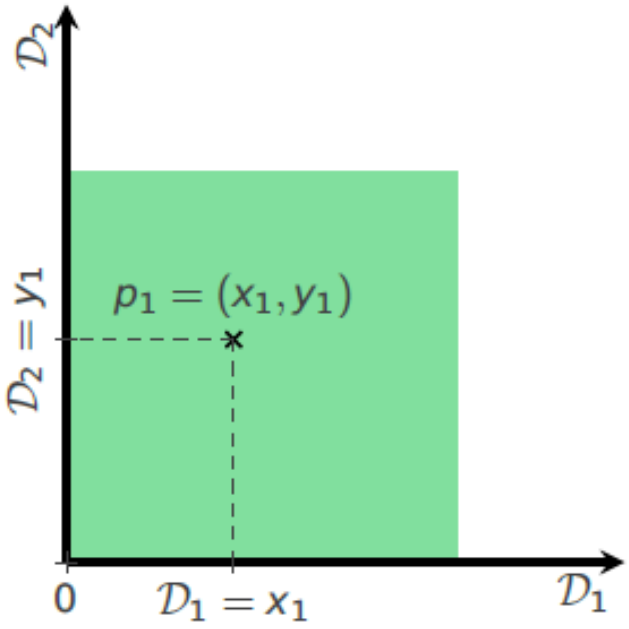
Problem components

- Each degree of freedom can be interpreted as an axis along which you can move.
- Each position on the axis hereby corresponds to a certain assignment of the degree of freedom.
- Together, the degrees of freedom span the so-called *search space* Ss of the problem.



Problem components

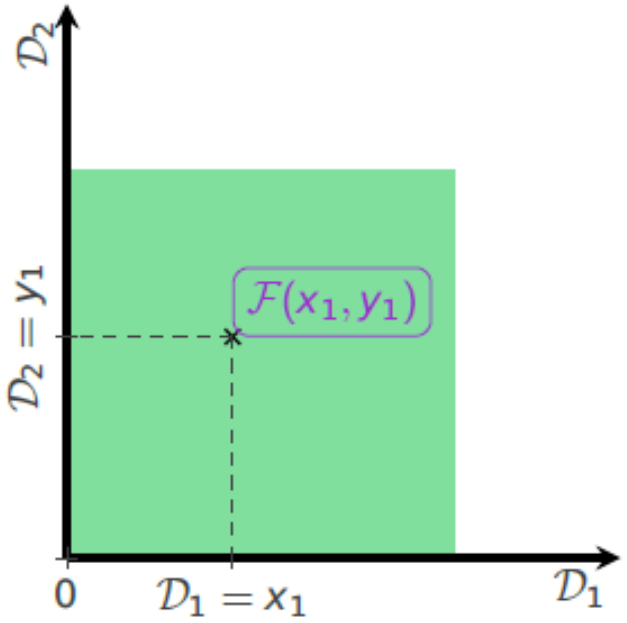
- Assigning a value to each degree of freedom defines
  a) a concrete problem solution and
  b) a point *p* within the search space



Problem components

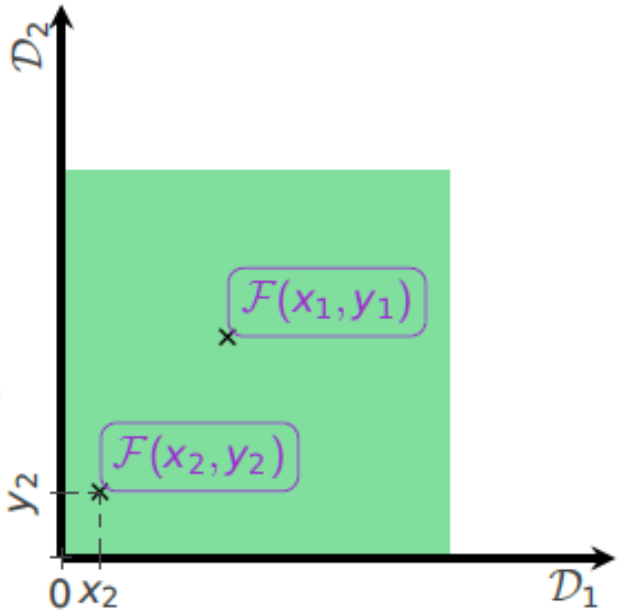
- Assigning a value to each degree of freedom defines
  a) a concrete problem solution and
  b) a point *p* within the search space

- Entering the coordinates of a search space point into the evaluation function yields the quality of the corresponding solution w.r.t. the corresponding design objective.



Problem components

- Comparing the quality, i.e., the corresponding output of the evaluation function(s), of multiple points allows to establish a *preference* between these points (and the corresponding problem solutions).



Preference

- Informal description:

Compared to problem solution A, problem solution B is preferable considering a design objective if the evaluation of B yields a value not greater than the value yielded when evaluating A.

- Formal definition:

Solution p_i is preferable to solution p_j considering the objective function $F(p)$ (denoted as $p_i \geq p_j$) if $F(p_i) \leq F(p_j)$:

$$p_i \geq p_j \Leftrightarrow F(p_i) \leq F(p_j)$$

Optimality

- Informal description:

A solution which is preferable to any other solution within a certain range/region is denoted as an *optimum* of the optimization problem. A solution which is preferable to any other solution within the entire search space is denoted as a *global optimum*. Any optimum which is not the global optimum is denoted as a *local optimum*.

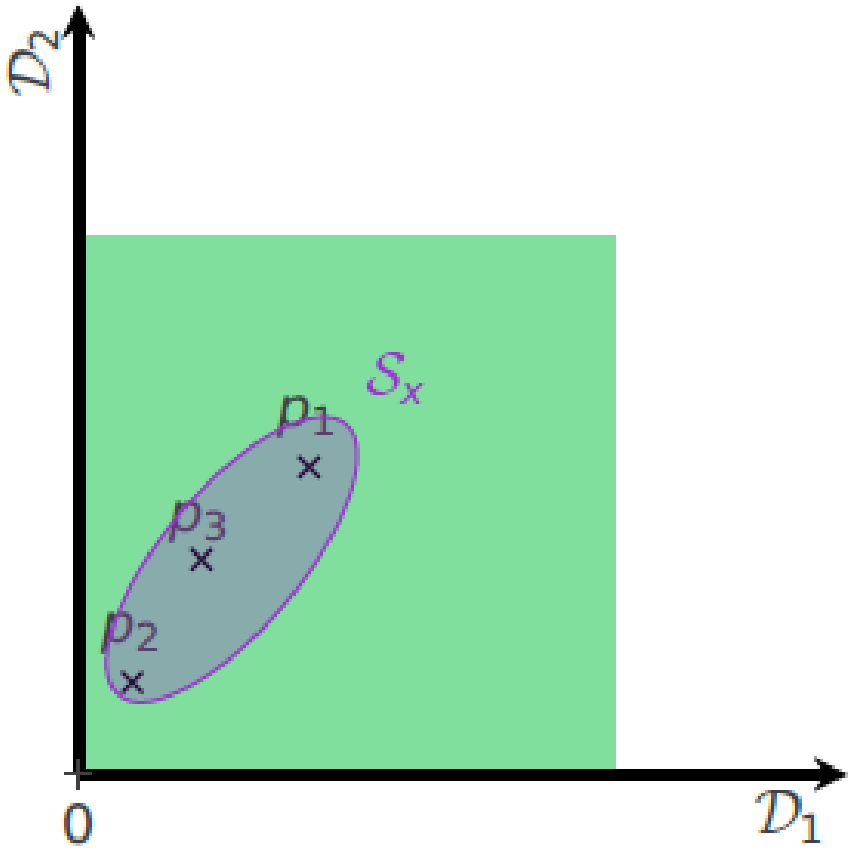
- Formal definition:

A solution $p_I \in S_s$ is considered as a local optimum of the space $S_x \subset S_s$ iff $p_I \geq p \forall\, p \in S_x$.
A solution $p_g \in S_s$ is considered as the global optimum iff $p_g \geq p \forall\, p \in S_s$.

Optimality

Example:

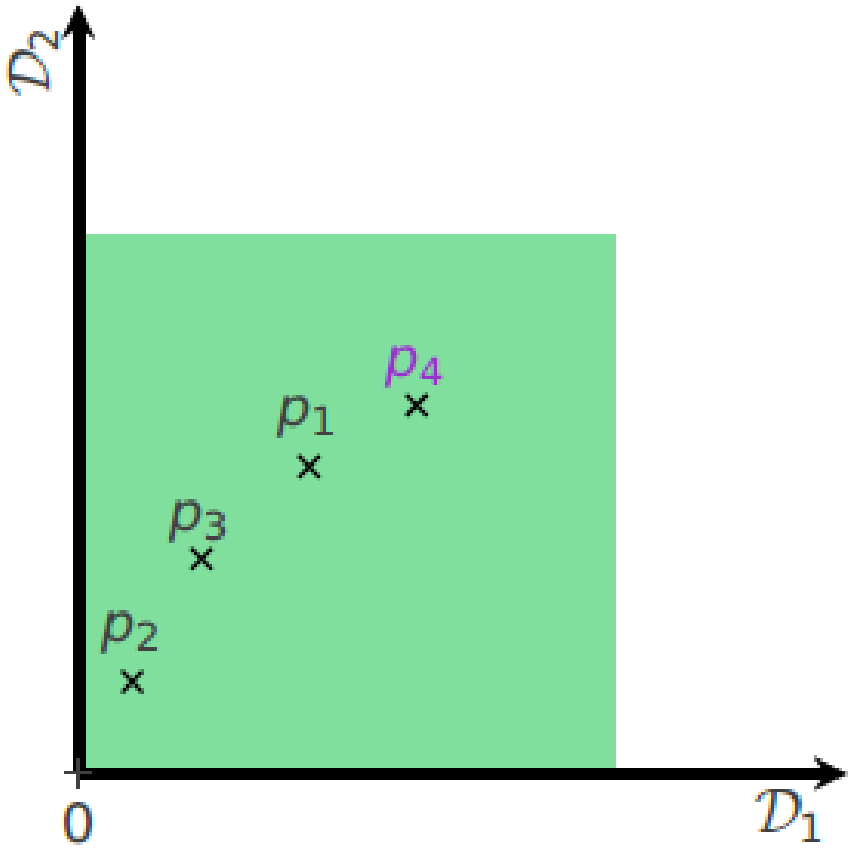
- p_3 is the local optimum within the region S_x if it is preferable to any other point in that region.



Optimality

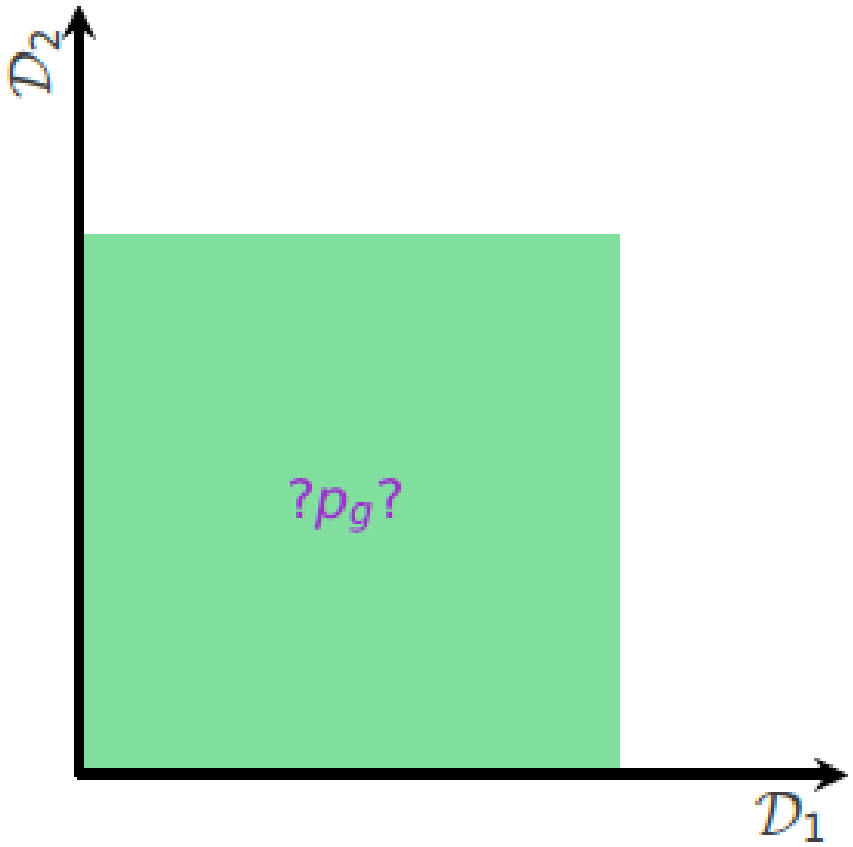
Example:

- p_3 is the local optimum within the region S_x if it is preferable to any other point in that region.
- p_4 is the global optimum of the problem if it is preferable to any other point in the entire search space.



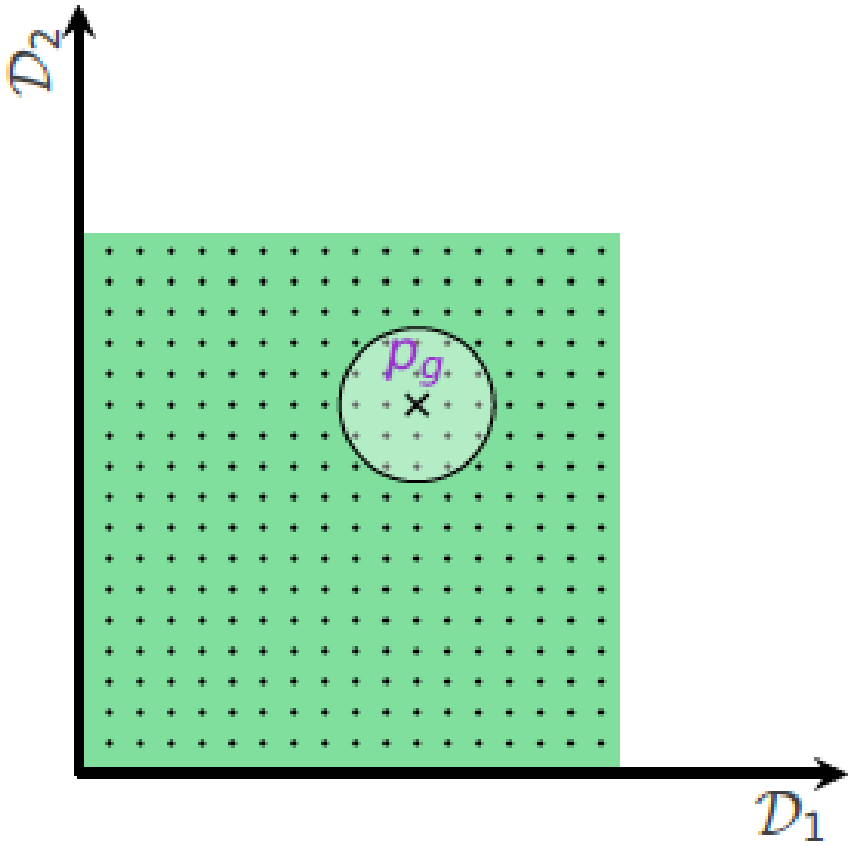
Optimality

- Finding the global optimum p_g within the search space S_s is the main goal during the optimization.



Search approaches

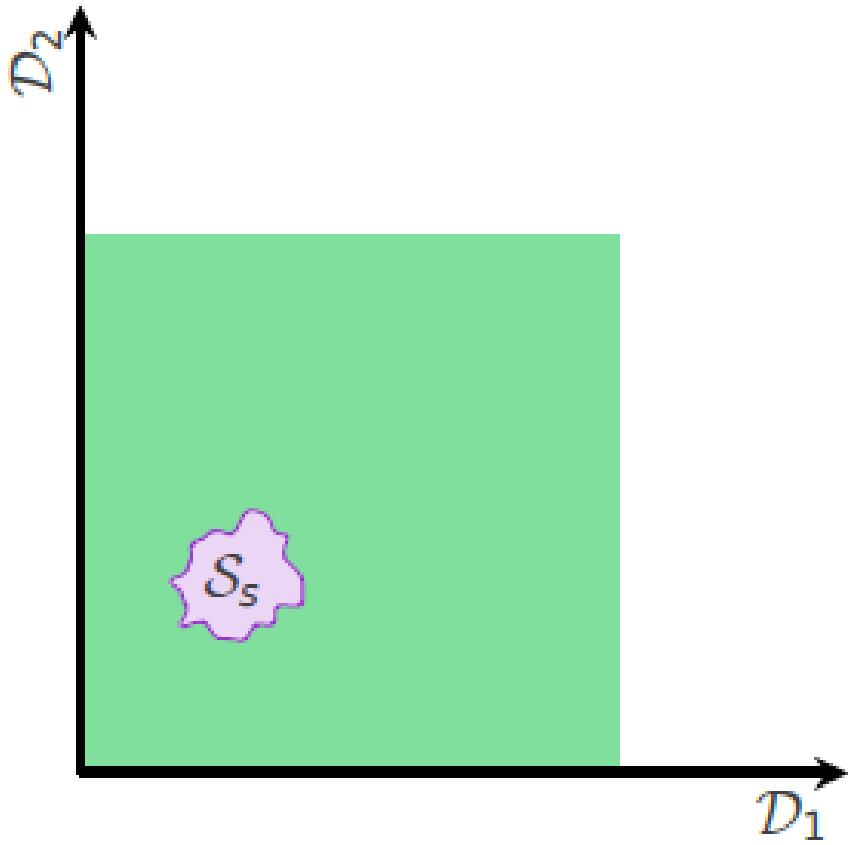
- Finding the global optimum p_g within the search space S_s is the main goal during the optimization.
- For small problem instances, the global optimum can be found by *exhaustively searching* through the search space (this is also denoted as *enumeration* or *brute force*).



Search or exploration hereby refers to an evaluation of the quality of a set of points considering the design objectives.

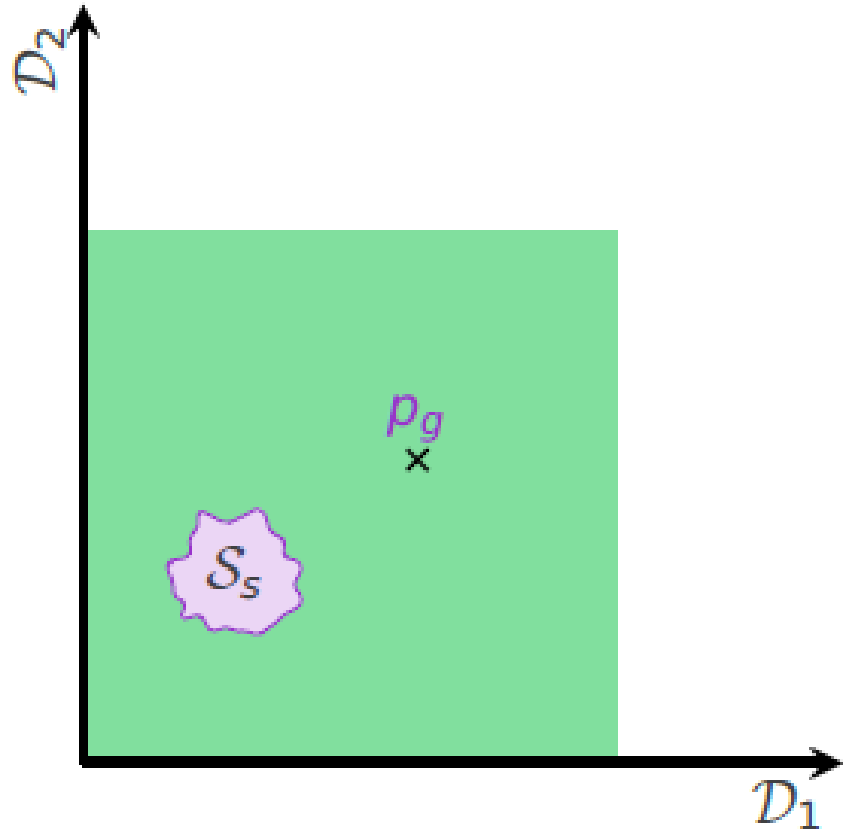
Search approaches

- For realistic problem sizes, an exhaustive search is impractical, if at all feasible.
- Practical problems have to be solved by exploring an exploration space $S_e \subset S_s$ which constitutes only a subspace of the search space S_s .



Search approaches

- For realistic problem sizes, an exhaustive search is impractical, if at all feasible.
- Practical problems have to be solved by exploring an exploration space $S_e \subset S_s$ which constitutes only a subspace of the search space S_s .

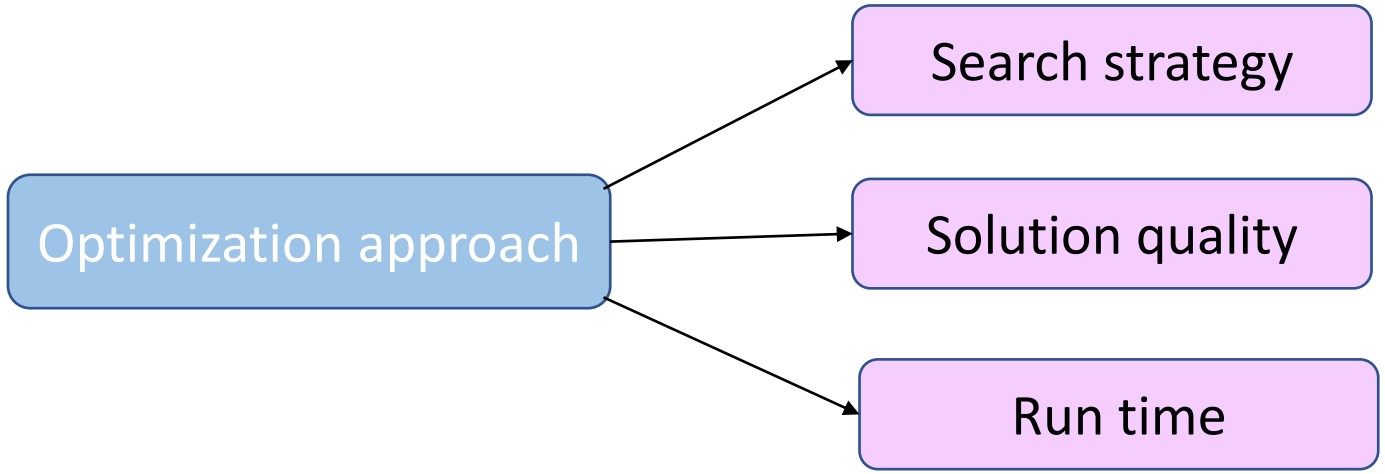


Searching only a subspace, while reducing the exploration time, naturally introduces the possibility of not finding the global optimum.

Search approaches

An optimization algorithm is characterized by...

- *search strategy:* the way it chooses the points for the evaluation during the exploration of the search space.
- *Efficiency:* the relation between its run time (given a certain compute power) and the quality of the found problem solutions (their proximity to the global optima).



Problem characteristic

Problem components

Optimization design decisions typically involve making choices to maximize or minimize certain objectives while satisfying specific constraints.

1. Assign *m* continuous and *n* binary **design decisions**:

$$D_c \in Q^m, D_b \in \{0,1\}^n$$

2. Subject to a set of **constraints**

$$a_c \,.\, a_b \leq C$$

3. So that a set of **objective functions** minimized or maximized.

$$F(D_c, D_b)$$

Overview of problem characteristics

- **Design decisions**:
  - Required quality
  - Dynamisms
  - Linearity
  - Number of objectives
  - Precision
- **Constraints:**
  - Design constraints
  - Objective constraints
  - Feasibility constraints
- **Objective functions:**
  - Dynamism
  - Precision
  - Discretization

Overview of problem characteristics

- **Design decisions**:
  - Required quality
  - Dynamisms
  - Linearity
  - Number of objectives
  - Precision
- **Constraints:**
  - Design constraints
  - Objective constraints
  - Feasibility constraints
- **Objective functions:**
  - Dynamism
  - Precision
  - Discretization

Required quality

Typically, realistic applications require a solution which is "good enough", rather than requiring the globally optimal solution. In some situations, being "better“ than a certain threshold does not provide practical benefits:

- Application makespan in relation to the user’s reaction speed.
- Typically, having more slack is not beneficial for a real-time application.
- An acceptable failure rate for applications (which are not safety-critical).

Considering their guarantees of finding the optimal solution, optimization algorithms are classified as...

- ***exact algorithms*** which, upon termination, are guaranteed to find the global optimum.
- ***Heuristic and metaheuristic algorithms*** which provide no such guarantees.

Exact optimization approaches

Exact optimization approaches

In addition to the Brute Force Enumeration, exact optimization algorithms encompass a type of algorithms with much higher practical relevance **guarantee** finding the optimal solution (within numerical precision) for well-defined optimization problems:

- Closed-Form *Approaches*
- Linear programming (ILPs, MILPs)
- Non-linear programming
- Convex optimization

Closed-Form approaches

A *closed-form solution* means you can write the optimal answer explicitly as a formula, usually using a finite number of standard operations.

The general functionality of these methods can be demonstrated with a trivial example.

- Assume that our optimization problem has the following objective function, which we want to minimize:

$$F(x) = x^4 + 2x^2 + 1$$

Closed-Form approaches

The general functionality of these methods can be demonstrated with a trivial example.

- Assume that our optimization problem has the following objective function, which we want to minimize:

$$F(x) = x^4 + 2x^2 + 1$$

- Examining the function equation immediately reveals that, regardless of the value of x, the values x^4 and $2x^2$ are always positive. Consequently, the minimum of the function is $F(x) = 1$ and the point $x = 0$ is the global optimum of the given optimization problem.

Indeed, we can go a step further by generalizing this argumentation to say that the minimum of any polynomial function where the degrees of freedom have (a) positive coefficients and (b) even exponents is found at $x = 0$.

Closed-Form approaches

- Examining the function equation immediately reveals that, regardless of the value of x, the values x^4 and $2x^2$ are always positive. Consequently, the minimum of the function is $F(x) = 1$ and the point $x = 0$ is the global optimum of the given optimization problem. Indeed, we can go a step further by generalizing this argumentation to say that the minimum of any polynomial function where the degrees of freedom have (a) positive coefficients and (b) even exponents is found at $x = 0$.

Closed-Form approaches are characterized by:

- A mathematical analysis of the optimized function to find...
- general function properties...
- to directly infer the location of the global optimum...
- and then generalize the findings to all functions fulfilling specific conditions.

Closed-Form approaches characteristics

Closed-Form approaches are characterized by:

- "*A **mathematical analysis** of the optimized function to find **general function properties** to directly (i.e., without the necessity to evaluate many concrete problem solutions) **infer** the location of the global optimum and then **generalize** the findings to all functions fulfilling **specific conditions**.*"

Closed-Form approaches...

- are **exact** approaches (they guarantee to find the global optimum).
- are often **computationally efficient** (given that they yield the global optimum).
- have strict conditions which can **limit their application** to realistic problems.

Closed-Form approaches

- The term closed-form also often refers to elegant and compact solutions of initially large and complex problems, such as constant-size formulas for the calculation of sequences, such as the ***Gauss sequence***:

$$\sum_{i=1}^{n} i = 1 + 2 + 3 + \cdots + n$$

- which has the closed-form solution:

$$\sum_{i=1}^{n} i = 0.6.n.(n+1)$$

Closed-Form approaches

"A closed-form solution (or closed-form expression) is any **formula** that can be evaluated in a finite number of standard operations." - From the blog of Glyn Holton.

- In the context of optimization, closed-form approaches refer to situations where it is possible to mathematically determine the global optimum.
- Problems with monotonic evaluation functions are typical practical examples of problems where closed-form solutions can be found.

However,

- There are many optimization problems where it is difficult of impossible to formulate a closed-form solution.
- Factors which make it difficult are, e.g., evaluation functions which are...
  - non linear.
  - non differentiable.

Closed-Form approaches

- Formulating closed-form solutions is particularly challenging for problems where it is already difficult to formulate a mathematical relation between the design decisions and the quality of the corresponding problem solution (e.g., the evaluation of system reliability).
- The fact that the evaluation functions of realistic problems are oftentimes not reversible further hinders the formulation of closed-form solutions.

Closed-Form approaches	
Advantages	Disadvantages
Exact approach	Difficult to apply for many practical problems
Oftentimes constant run time	Single objective

Exact optimization approaches-ILPs

- ILPs...
- are **exact** optimizers.
- consider the entire search space (given by the set of encoding variables).
- require **linearity**, both for the constraint inequalities and the objective function.
- are solved using specialized ***backtracking, branch and bound, branch and cut, …*** solvers

Exact optimization approaches-ILPs

Backtracking is a special form of enumeration. The goal is to perform a search of the search space which is...

- **Exhaustive**, i.e., the search considers **each search space point**.
- **Efficient**, i.e., the search considers each search space point **exactly once**.

Typical applications of backtracking include...

- Finding **all subsets** of a given set of *n* objects (search space size 2*n*)
- Finding **all permutations** of a list of *n* objects (search space size *n*!)
- Finding **all (cycle-free) paths** between two nodes in a graph with *n* nodes (search space size depends on the graph topology)

Exact optimization approaches-ILPs

- Backtracking algorithms are typically based on...
  - The establishment of an **order** between the problem components
  - A **constructive creation** of a solution over multiple iterations, where one problem component of the current iteration is determined in each iteration

Due to their complexity, **general backtracking problems** can only be solved for problems with a size $n = 60$. However, the usage of **pruning techniques** makes it possible to apply backtracking to much larger (constrained) optimization problems.

Exact optimization approaches-ILPs

Pruning techniques are used to...

- **exclude a set of points** (a search space area) from the search space, i.e., determine that none of these points are the optimal solution of the program...
- based on insights derived from a **partial solution** of the problem.

Pruning is integrated into backtracking approaches by introducing **stopping conditions** which are checked during every step of the solution construction.

- The effectiveness of pruning approaches significantly depends on...
- the **overhead** of evaluating the stopping conditions.
- the **order** of the solution construction steps.

Optimality

- Informal description:

Integer Linear Programming (ILP) problem is an optimization (or a feasibility problem) which is defined by a **linear** objective function and a set of **linear** constraints. Both the constraints and the objective function are formulated based on (0-1) integer variables.

- Formal definition:

1. A (0-1) ILP problem is defined by an objective function:

$$\text{minimize } C^T x$$

2. subject to a set of constraints:

$$Ax \leq b$$

3. The problem is resolved by assigning values to a set of *n* (0-1) variables:

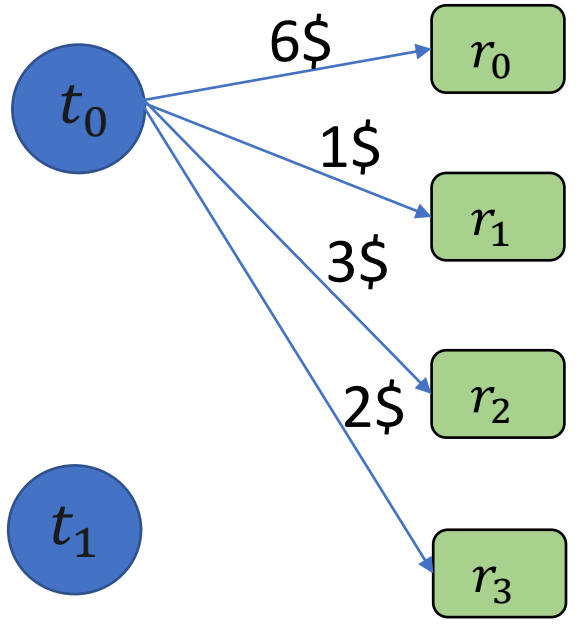
$$x \in \{0,1\}^n$$

Exact optimization approaches-ILPs

Example:

Assigning two tasks onto 4 available resources so that:

- each task is assigned to at least one resource...
- the tasks are not assigned to the same resource...
- and the overall costs are minimal.

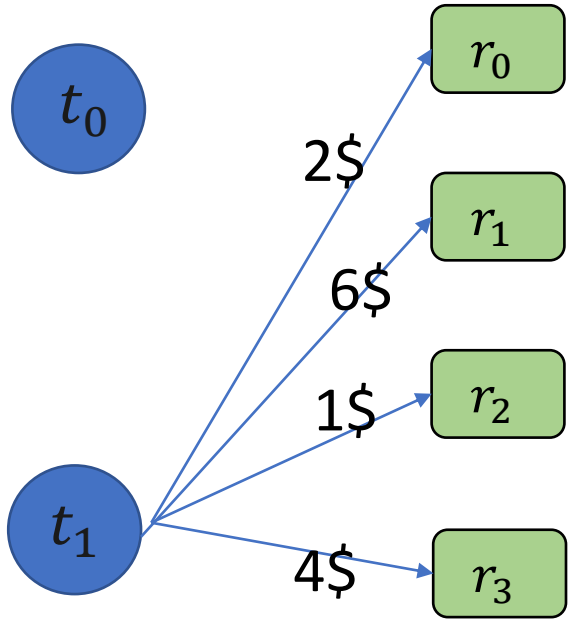


Exact optimization approaches-ILPs

Example:

Assigning two tasks onto 4 available resources so that:

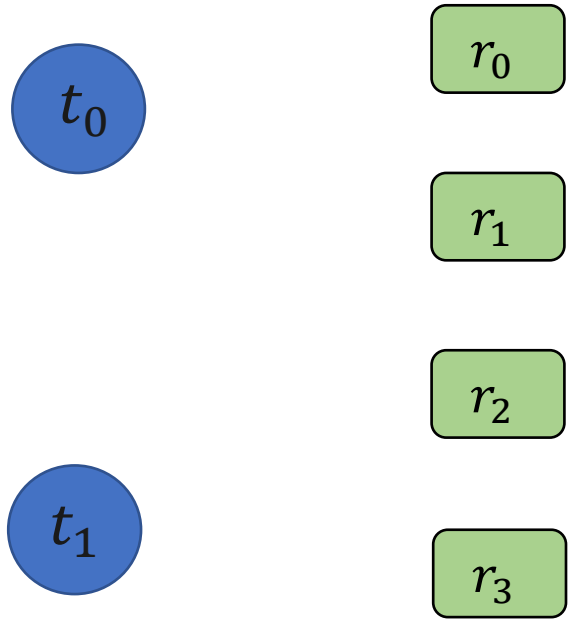
- each task is assigned to at least one resource...
- the tasks are not assigned to the same resource...
- and the overall costs are minimal.



Exact optimization approaches-ILPs

Definition of the variables:

- Each encoding variable $\tilde{m} \in V$ is set to either 1 or 0.
- Hereby, the variable $\tilde{m}_{ij}$ is set to 1 if the task t_i is executed on resource r_j.

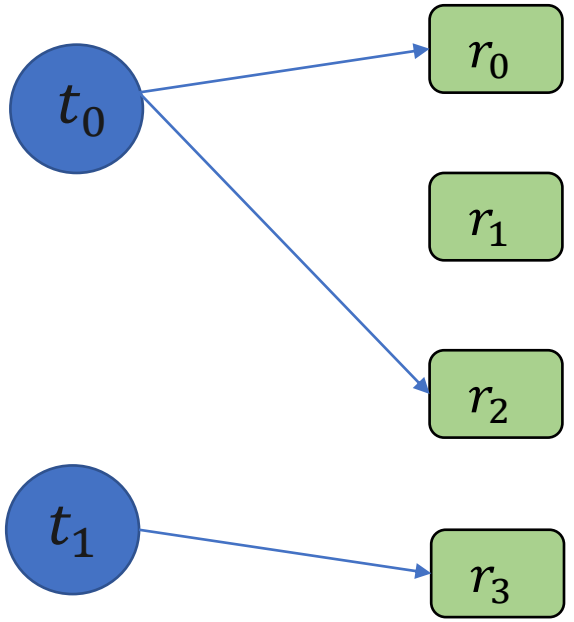


Exact optimization approaches-ILPs

Definition of the variables:

- Each encoding variable $\tilde{m} \in V$ is set to either 1 or 0.
- Hereby, the variable $\tilde{m}_{ij}$ is set to 1 iff the task t_i is executed on resource r_j.
- For example, executing task t_0 on the resources r_0 and r_2 and task t_1 on resource r_3 corresponds to the variable vector x with:

$$x^T = (\underset{\tilde{m}_{00}}{1},\underset{\tilde{m}_{01}}{0},\underset{\tilde{m}_{02}}{1},\underset{\tilde{m}_{03}}{0},\underset{\tilde{m}_{10}}{0},\underset{\tilde{m}_{11}}{0},\underset{\tilde{m}_{12}}{0},\underset{\tilde{m}_{13}}{1})$$



Exact optimization approaches-ILPs

Formulating the ILP system:

- A task is executed on at least one resource (Eqs. 1 and 2).

$$\tilde{m}_{00} + \tilde{m}_{01} + \tilde{m}_{02} + \tilde{m}_{03} \geq 1 \quad (1)$$

$$\tilde{m}_{10} + \tilde{m}_{11} + \tilde{m}_{12} + \tilde{m}_{13} \geq 1 \quad (2)$$

Exact optimization approaches-ILPs

Formulating the ILP system:

- A task is executed on at least one resource (Eqs. 1 and 2).
- The tasks are not assigned to the same resource (Eqs. 3–6).

$$\tilde{m}_{00} + \tilde{m}_{01} + \tilde{m}_{02} + \tilde{m}_{03} \geq 1 \quad (1)$$

$$\tilde{m}_{10} + \tilde{m}_{11} + \tilde{m}_{12} + \tilde{m}_{13} \geq 1 \quad (2)$$

$$\tilde{m}_{00} + \tilde{m}_{10} \leq 1 \quad (3)$$

$$\tilde{m}_{01} + \tilde{m}_{11} \leq 1 \quad (4)$$

$$\tilde{m}_{02} + \tilde{m}_{12} \leq 1 \quad (5)$$

$$\tilde{m}_{03} + \tilde{m}_{13} \leq 1 \quad (6)$$

Exact optimization approaches-ILPs

Formulating the ILP system:

- A task is executed on at least one resource (Eqs. 1 and 2).
- The tasks are not assigned to the same resource (Eqs. 3–6).
- The overall costs *X* are to be minimized (Eq. 7).

$$\tilde{m}_{00} + \tilde{m}_{01} + \tilde{m}_{02} + \tilde{m}_{03} \geq 1 \quad (1)$$

$$\tilde{m}_{10} + \tilde{m}_{11} + \tilde{m}_{12} + \tilde{m}_{13} \geq 1 \quad (2)$$

$$\tilde{m}_{00} + \tilde{m}_{10} \leq 1 \quad (3)$$

$$\tilde{m}_{01} + \tilde{m}_{11} \leq 1 \quad (4)$$

$$\tilde{m}_{02} + \tilde{m}_{12} \leq 1 \quad (5)$$

$$\tilde{m}_{03} + \tilde{m}_{13} \leq 1 \quad (6)$$

$$6\tilde{m}_{00} + \tilde{m}_{01} + 3\tilde{m}_{02} + 2\tilde{m}_{03} + 2\tilde{m}_{10} + 5\tilde{m}_{11} + \tilde{m}_{12} + 4\tilde{m}_{13} = x \quad (7)$$

Exact optimization approaches-ILPs

$$Ax \leq b$$

$$\boldsymbol{A} = \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Constraints

$$\tilde{m}_{00} + \tilde{m}_{01} + \tilde{m}_{02} + \tilde{m}_{03} \geq 1 \quad (1)$$

$$\tilde{m}_{10} + \tilde{m}_{11} + \tilde{m}_{12} + \tilde{m}_{13} \geq 1 \quad (2)$$

$$\tilde{m}_{00} + \tilde{m}_{10} \leq 1 \quad (3)$$

$$\tilde{m}_{01} + \tilde{m}_{11} \leq 1 \quad (4)$$

$$\tilde{m}_{02} + \tilde{m}_{12} \leq 1 \quad (5)$$

$$\tilde{m}_{03} + \tilde{m}_{13} \leq 1 \quad (6)$$

Design decisions

$$\boldsymbol{x}^\top = \begin{bmatrix} \tilde{m}_{00} & \tilde{m}_{01} & \tilde{m}_{02} & \tilde{m}_{03} & \tilde{m}_{10} & \tilde{m}_{11} & \tilde{m}_{12} & \tilde{m}_{13} \end{bmatrix}$$

$$\boldsymbol{b}^\top = \begin{bmatrix} -1 & -1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Exact optimization approaches-ILPs

$$\text{minimize } c^\top x$$

$$c^\top = \begin{bmatrix} 6 & 1 & 3 & 2 & 2 & 5 & 1 & 4 \end{bmatrix}$$

$$x = \begin{bmatrix} \tilde{m}_{00} \\ \tilde{m}_{01} \\ \tilde{m}_{02} \\ \tilde{m}_{03} \\ \tilde{m}_{10} \\ \tilde{m}_{11} \\ \tilde{m}_{12} \\ \tilde{m}_{13} \end{bmatrix}$$

Objective Function

$$6\tilde{m}_{00} + \tilde{m}_{01} + 3\tilde{m}_{02} + 2\tilde{m}_{03} + 2\tilde{m}_{10} + 5\tilde{m}_{11} + \tilde{m}_{12} + 4\tilde{m}_{13} = x \qquad (7)$$

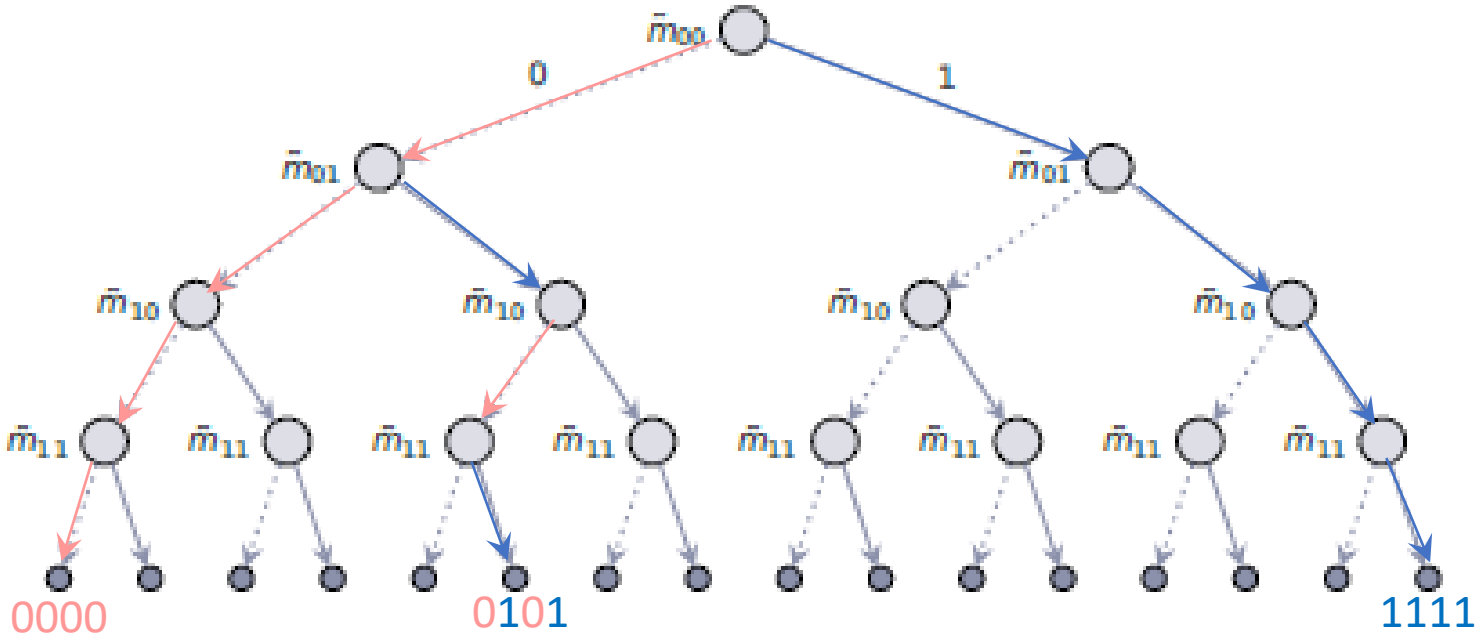
Problem Characteristics

For the sake of a clear visualization, in the following, we will only consider the task mapping on 2 out of the 4 resources.

$$x = \begin{bmatrix} \tilde{m}_{00} \\ \tilde{m}_{01} \\ \tilde{m}_{02} \\ \tilde{m}_{03} \\ \tilde{m}_{10} \\ \tilde{m}_{11} \\ \tilde{m}_{12} \\ \tilde{m}_{13} \end{bmatrix}$$

Problem Characteristics

$$x = \begin{bmatrix} \tilde{m}_{00} \\ \tilde{m}_{01} \\ 0 \\ 0 \\ \tilde{m}_{10} \\ \tilde{m}_{11} \\ 0 \\ 0 \end{bmatrix}$$



Problem characteristics

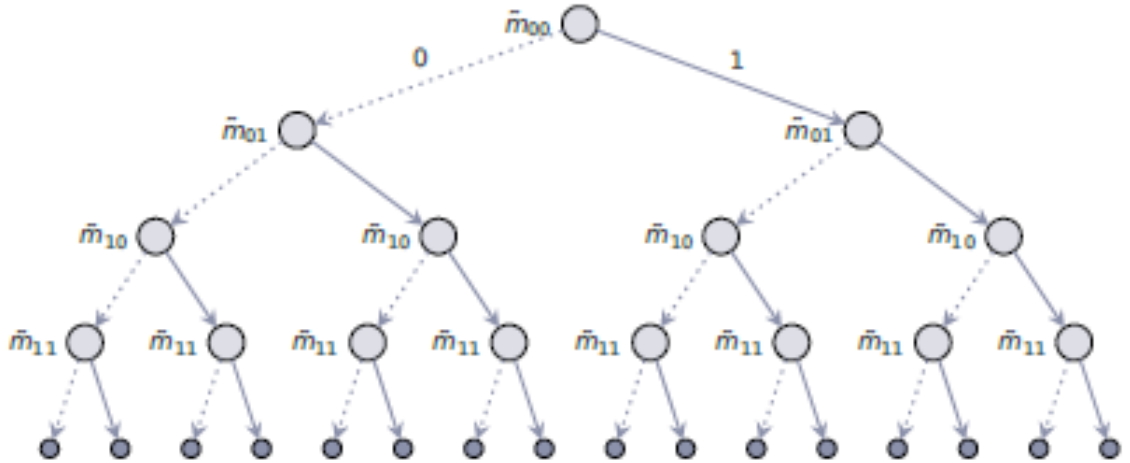
constraints

$$\tilde{m}_{00} + \tilde{m}_{01} \geq 1 \qquad (3)$$

$$\tilde{m}_{10} + \tilde{m}_{11} \geq 1 \qquad (4)$$

$$\tilde{m}_{00} + \tilde{m}_{10} \leq 1 \qquad (5)$$

$$\tilde{m}_{01} + \tilde{m}_{11} \leq 1 \qquad (6)$$



Objective Function

$$6\tilde{m}_{00} + \tilde{m}_{01} + 2\tilde{m}_{10} + 5\tilde{m}_{11} = x \qquad (7)$$

Problem characteristics

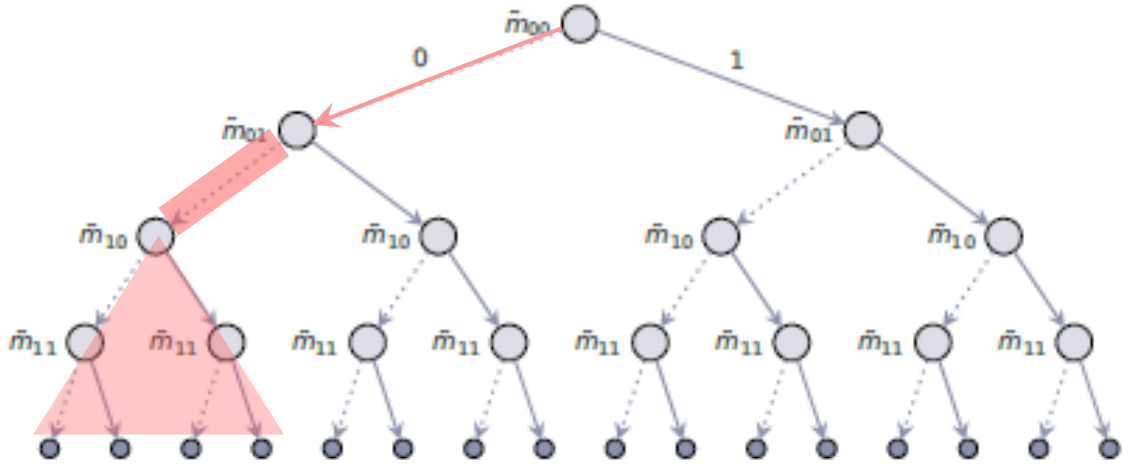
constraints

$$\tilde{m}_{00} + \tilde{m}_{01} \geq 1 \qquad (3)$$

$$\tilde{m}_{10} + \tilde{m}_{11} \geq 1 \qquad (4)$$

$$\tilde{m}_{00} + \tilde{m}_{10} \leq 1 \qquad (5)$$

$$\tilde{m}_{01} + \tilde{m}_{11} \leq 1 \qquad (6)$$



Objective Function

$$6\tilde{m}_{00} + \tilde{m}_{01} + 2\tilde{m}_{10} + 5\tilde{m}_{11} = x \qquad (7)$$

Problem Characteristics

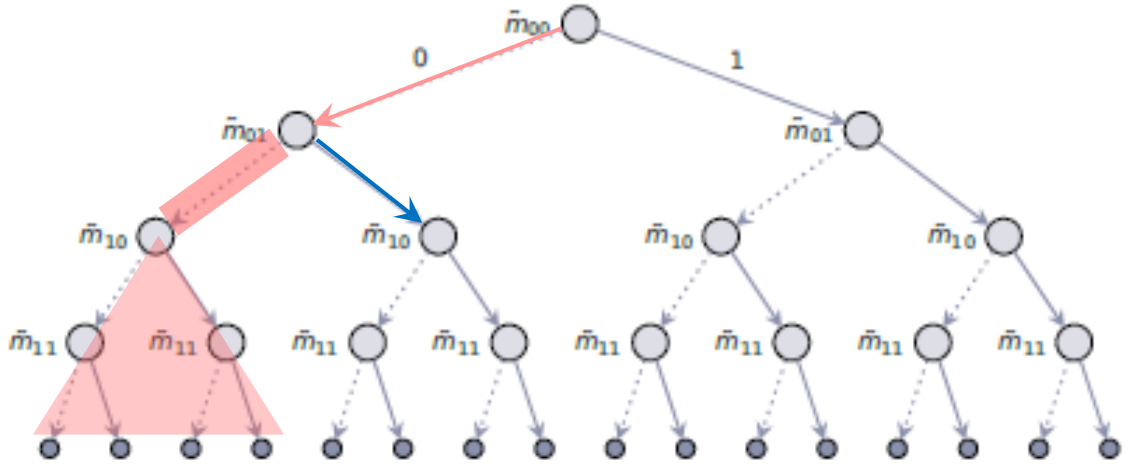
constraints

$$\tilde{m}_{00} + \tilde{m}_{01} \geq 1 \qquad (3)$$

$$\tilde{m}_{10} + \tilde{m}_{11} \geq 1 \qquad (4)$$

$$\tilde{m}_{00} + \tilde{m}_{10} \leq 1 \qquad (5)$$

$$\tilde{m}_{01} + \tilde{m}_{11} \leq 1 \qquad (6)$$



Objective Function

$$6\tilde{m}_{00} + \tilde{m}_{01} + 2\tilde{m}_{10} + 5\tilde{m}_{11} = x \qquad (7)$$

Problem Characteristics

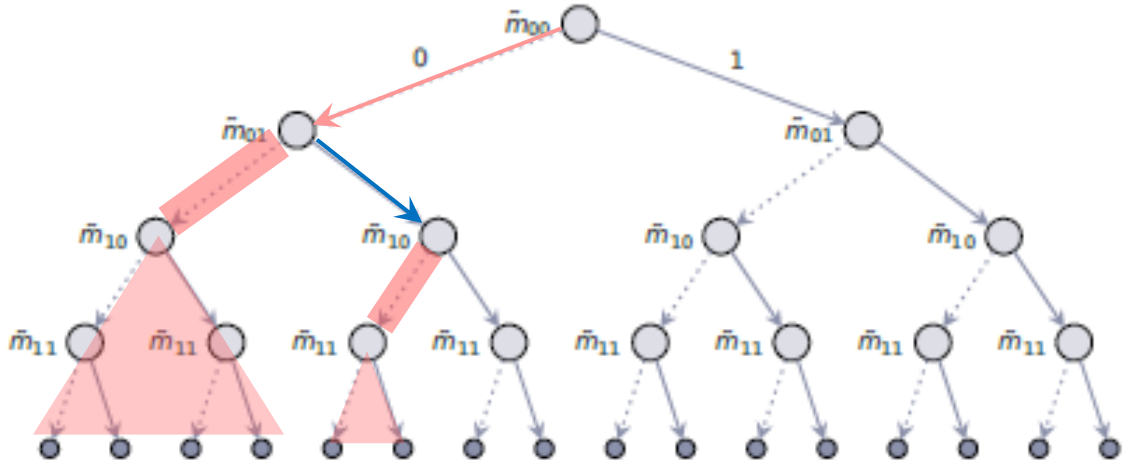
constraints

$$\tilde{m}_{00} + \tilde{m}_{01} \geq 1 \qquad (3)$$

$$\tilde{m}_{10} + \tilde{m}_{11} \geq 1 \qquad (4)$$

$$\tilde{m}_{00} + \tilde{m}_{10} \leq 1 \qquad (5)$$

$$\tilde{m}_{01} + \tilde{m}_{11} \leq 1 \qquad (6)$$



Objective Function

$$6\tilde{m}_{00} + \tilde{m}_{01} + 2\tilde{m}_{10} + 5\tilde{m}_{11} = x \qquad (7)$$

Problem Characteristics

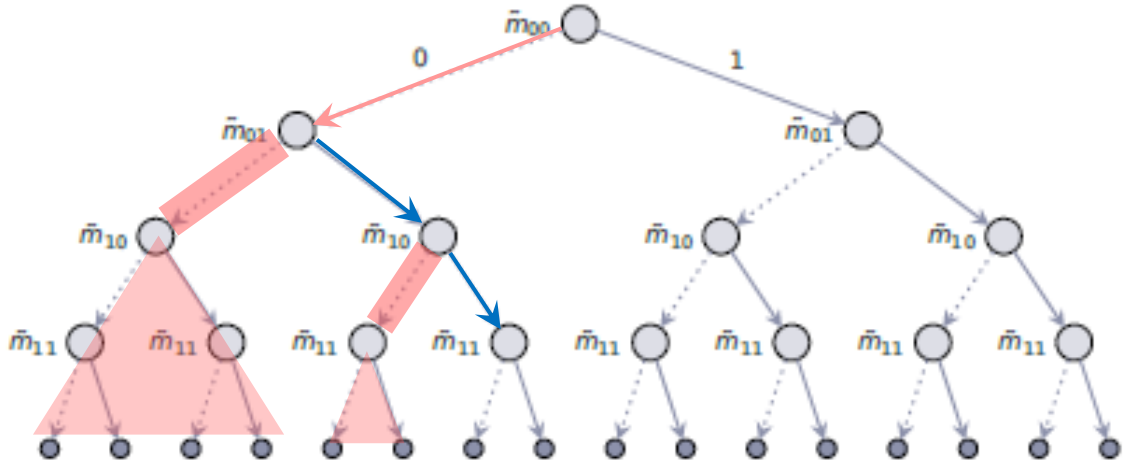
constraints

$$\tilde{m}_{00} + \tilde{m}_{01} \geq 1 \qquad (3)$$

$$\tilde{m}_{10} + \tilde{m}_{11} \geq 1 \qquad (4)$$

$$\tilde{m}_{00} + \tilde{m}_{10} \leq 1 \qquad (5)$$

$$\tilde{m}_{01} + \tilde{m}_{11} \leq 1 \qquad (6)$$



Objective Function

$$6\tilde{m}_{00} + \tilde{m}_{01} + 2\tilde{m}_{10} + 5\tilde{m}_{11} = x \qquad (7)$$

Problem Characteristics

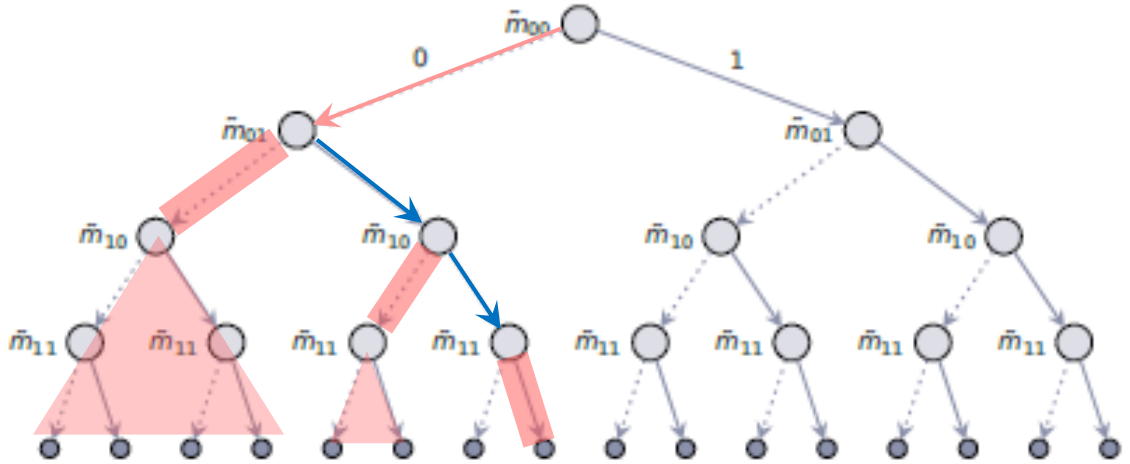
constraints

$$\tilde{m}_{00} + \tilde{m}_{01} \geq 1 \qquad (3)$$

$$\tilde{m}_{10} + \tilde{m}_{11} \geq 1 \qquad (4)$$

$$\tilde{m}_{00} + \tilde{m}_{10} \leq 1 \qquad (5)$$

$$\tilde{m}_{01} + \tilde{m}_{11} \leq 1 \qquad (6)$$



Objective Function

$$6\tilde{m}_{00} + \tilde{m}_{01} + 2\tilde{m}_{10} + 5\tilde{m}_{11} = x \qquad (7)$$

Problem Characteristics

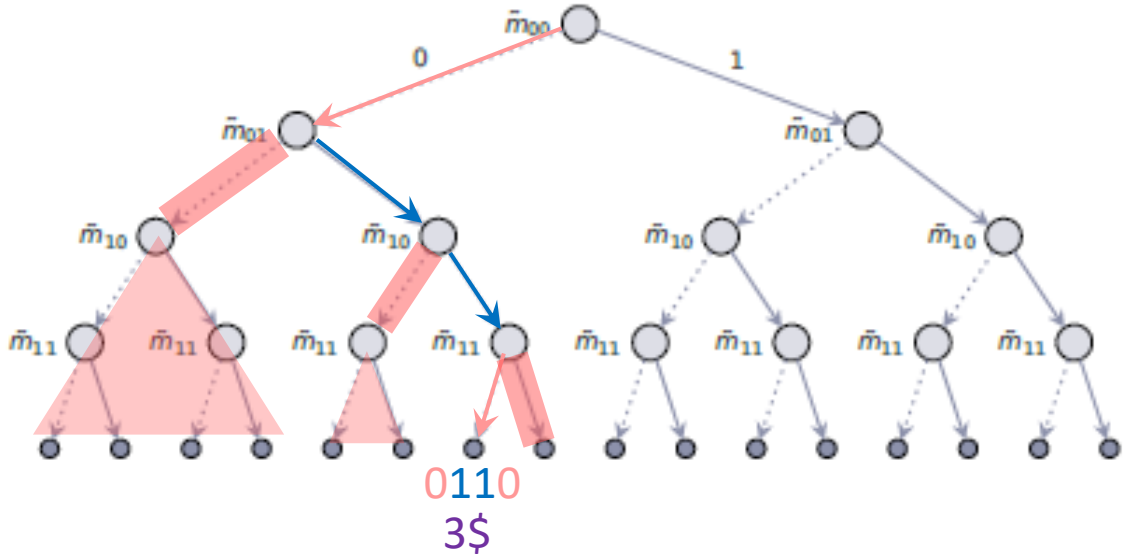
constraints

$$\tilde{m}_{00} + \tilde{m}_{01} \geq 1 \qquad (3)$$

$$\tilde{m}_{10} + \tilde{m}_{11} \geq 1 \qquad (4)$$

$$\tilde{m}_{00} + \tilde{m}_{10} \leq 1 \qquad (5)$$

$$\tilde{m}_{01} + \tilde{m}_{11} \leq 1 \qquad (6)$$



Objective Function

$$6\tilde{m}_{00} + \tilde{m}_{01} + 2\tilde{m}_{10} + 5\tilde{m}_{11} = x \qquad (7)$$

$$6 \cdot 0 + 1 \cdot 1 + 2 \cdot 1 + 5 \cdot 0 = 3$$

Problem Characteristics

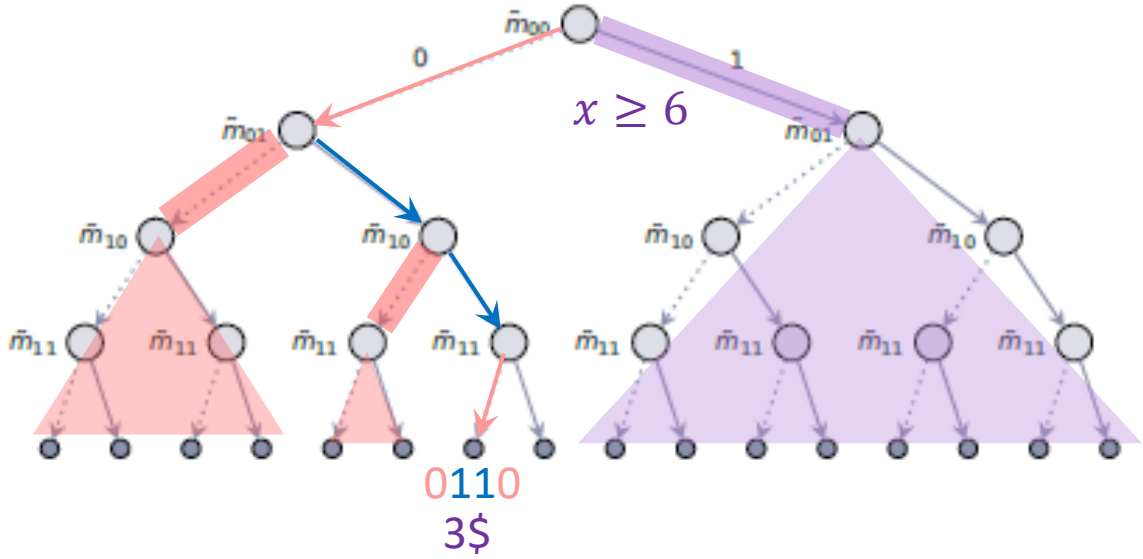
constraints

$$\tilde{m}_{00} + \tilde{m}_{01} \geq 1 \qquad (3)$$

$$\tilde{m}_{10} + \tilde{m}_{11} \geq 1 \qquad (4)$$

$$\tilde{m}_{00} + \tilde{m}_{10} \leq 1 \qquad (5)$$

$$\tilde{m}_{01} + \tilde{m}_{11} \leq 1 \qquad (6)$$



Objective Function

$$6\tilde{m}_{00} + \tilde{m}_{01} + 2\tilde{m}_{10} + 5\tilde{m}_{11} = x \qquad (7)$$

Exact optimization approaches-ILPs

ILPs...

- are **exact** optimizers.
- consider the entire search space (given by the set of encoding variables).
- require **linearity**, both for the constrain inequalities and the objective function.

The exploration of the search space...

- corresponds to the traversal of a binary tree.
- is done by specialized, highly sophisticated solvers.
- significantly reduces the area of the explored space by means of pruning and backtracking techniques.
- requires a time which scales exponentially with the number of the search space dimensions, i.e., the number of encoding variables.

Exact optimization approaches-ILPs

- Exact approaches are guaranteed to find the optimal solution. Custom designed solvers enable the solution of large realistic problems with comparatively little effort.
- Consequently, exact approaches are **the only true way** of solving optimization problems.
- However…
  a) formulating a problem within the framework is difficult and not always possible.
  b) It still run into scalability problems.

ILPS	
Advantages	Disadvantages
Exact approach	Scale exponentially
Significant pruning potential	Single objective
	Require linear function

Heuristic approaches

Heuristic approaches

Heuristic approaches are tailored to the problem at hand, problem-specific heuristics provide **near-optimal results** after **short** computation times (oftentimes in real time). Consequently, they are considered practically relevant in situations where quick decision-making is essential.

However,..

- the performance of problem-specific heuristics is based on **hard assumptions** about the problem (a deliberate narrowing of the search space) and is
  a) **achieved** only if the problem **exactly fits** to these assumptions.
  b) **maintained** only as long as the problem **does not change**.

Scheduling heuristics

Due to the high practical relevance of the scheduling of systems, there are many well-known scheduling heuristics such as:

- First-Come-First-Served (FCFS)
- Round-Robin (RR)
- Priority-Based (PB)
- Heterogeneous Earliest Finish Time (HEFT)

Generally speaking, all of these algorithms share the following traits:

- The tasks are scheduled based on a **given prioritization**
- The scheduling decisions are made **sequentially**
- The decision making algorithms are **greedy** and consider the currently scheduled task(s) in isolation

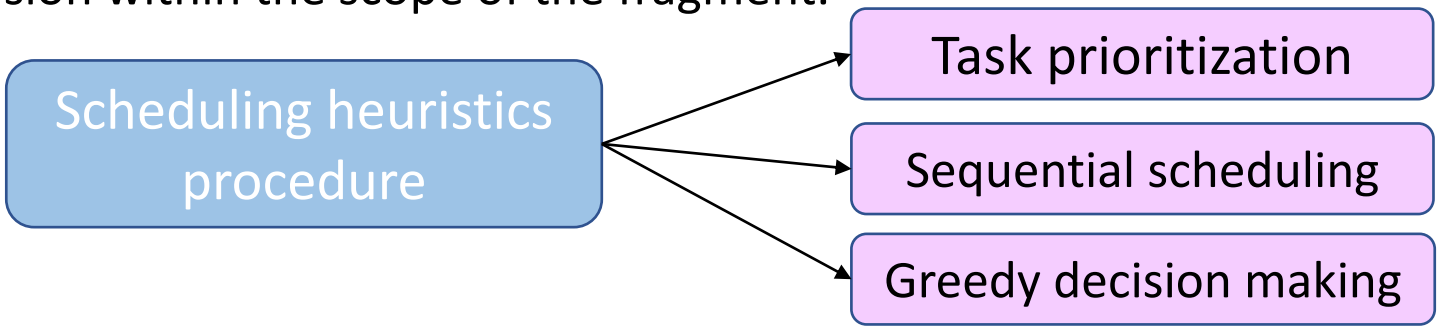
Scheduling heuristics

On a more general level, optimization heuristics address problems by...

- dividing the problem into isolated fragments...
- sorting these fragments according to a prioritization criterion,...
- and sequentially processing them following the prioritization order.

During the processing of each problem fragment, heuristics...

- consider only the problem characteristics within the scope of the fragment.
- greedily pick the best decision within the scope of the fragment.



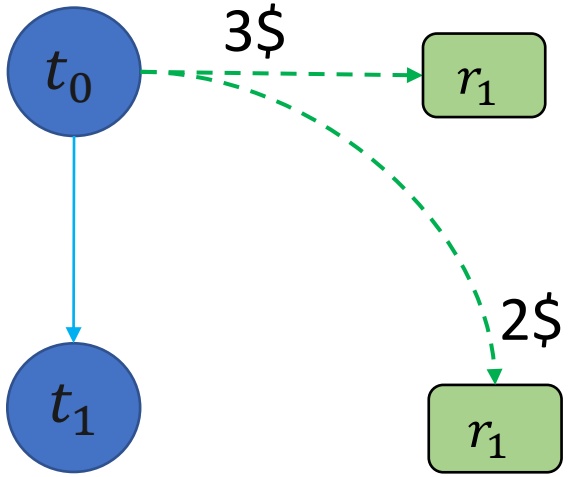
Scheduling heuristics

Consequently, heuristic approaches...

- achieve a high performance by significantly reducing the search space size (the dimensionality of the fragments is much lower than that of the whole problem).
- when making decisions for a fragment, are blind to the implications of these decisions for the fragments which are processed later.
- not capable of accepting a locally suboptimal decision to obtain a globally optimal solution down the line.

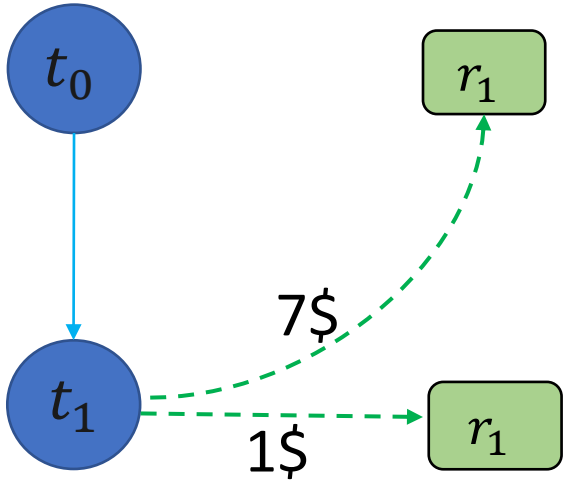
Scheduling heuristics

Example heuristic for cost-efficient scheduling:



Scheduling heuristics

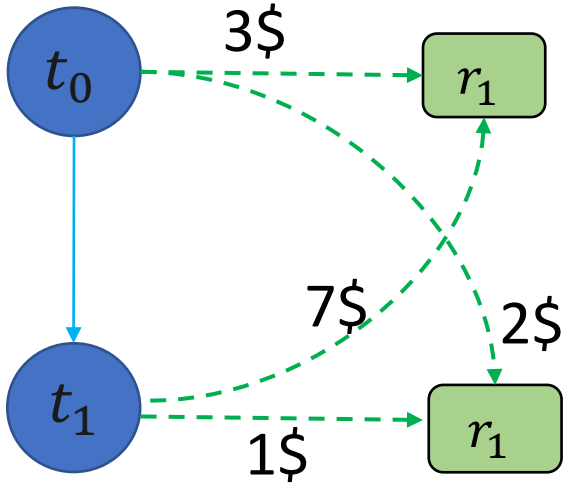
Example heuristic for cost-efficient scheduling:



Scheduling heuristics

Example heuristic for cost-efficient scheduling:

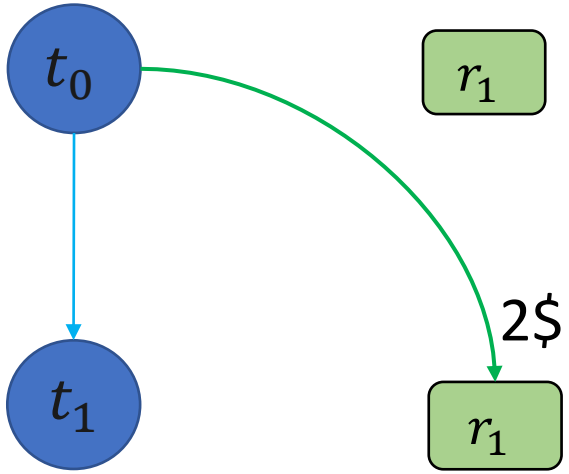
- Prioritization according to the data dependencies
- Assignment on the cheapest resource which is available



Scheduling heuristics

Example heuristic for cost-efficient scheduling:

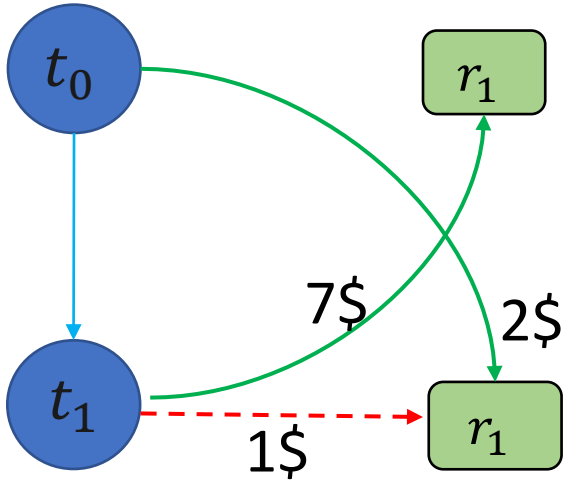
- Prioritization according to the data dependencies
- Assignment on the cheapest resource which is available
- The mapping of each task constitutes a problem fragment.



Scheduling heuristics

Example heuristic for cost-efficient scheduling:

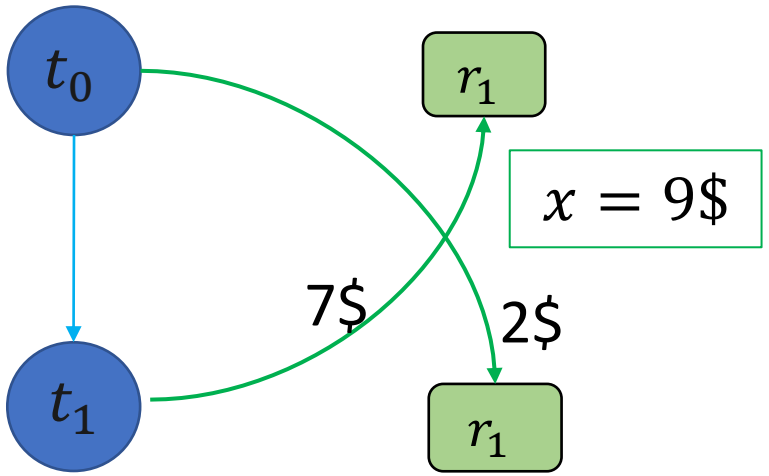
- Prioritization according to the data dependencies
- Assignment on the cheapest resource which is available
- The mapping of each task constitutes a problem fragment.



Scheduling heuristics

Example heuristic for cost-efficient scheduling:

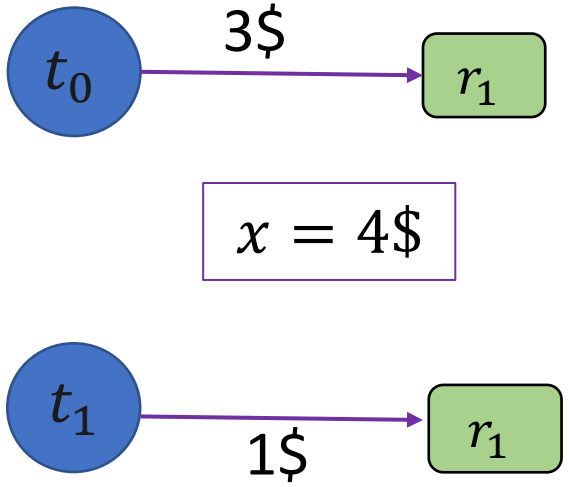
- Prioritization according to the data dependencies
- Assignment on the cheapest resource which is available
- The mapping of each task constitutes a problem fragment.



Scheduling heuristics

Example heuristic for cost-efficient scheduling:

- Prioritization according to the data dependencies
- Assignment on the cheapest resource which is available
- The mapping of each task constitutes a problem fragment.
- Limiting the scope to individual tasks and may exclude global optima.

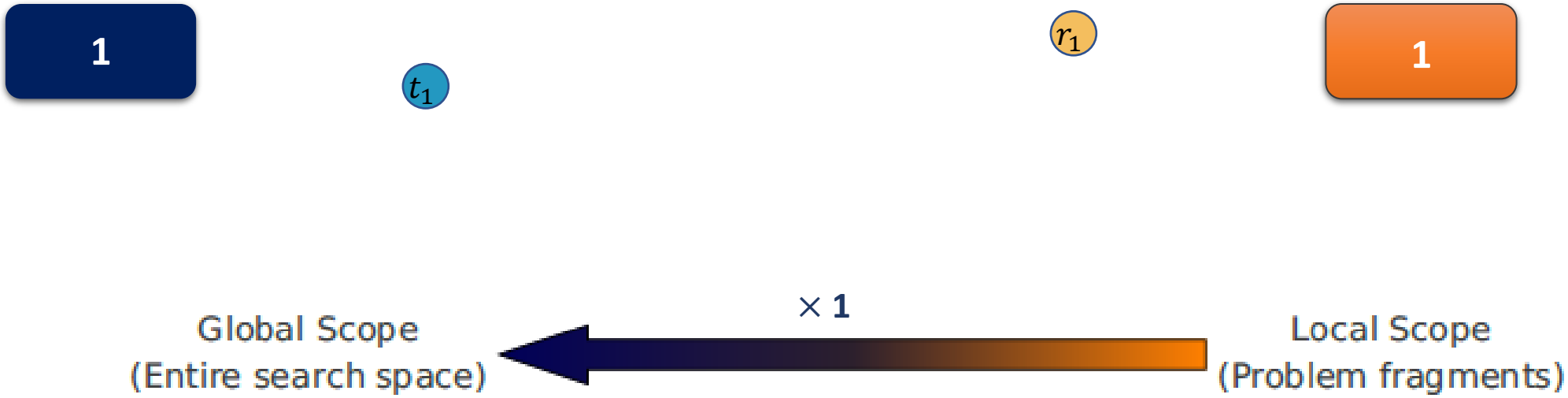


Scheduling heuristics

- Limiting the scope to problem fragments results in optimization speedup which scales with the problem size:

Number of necessary evaluations:

Mapping 1 tasks onto 1 resources.

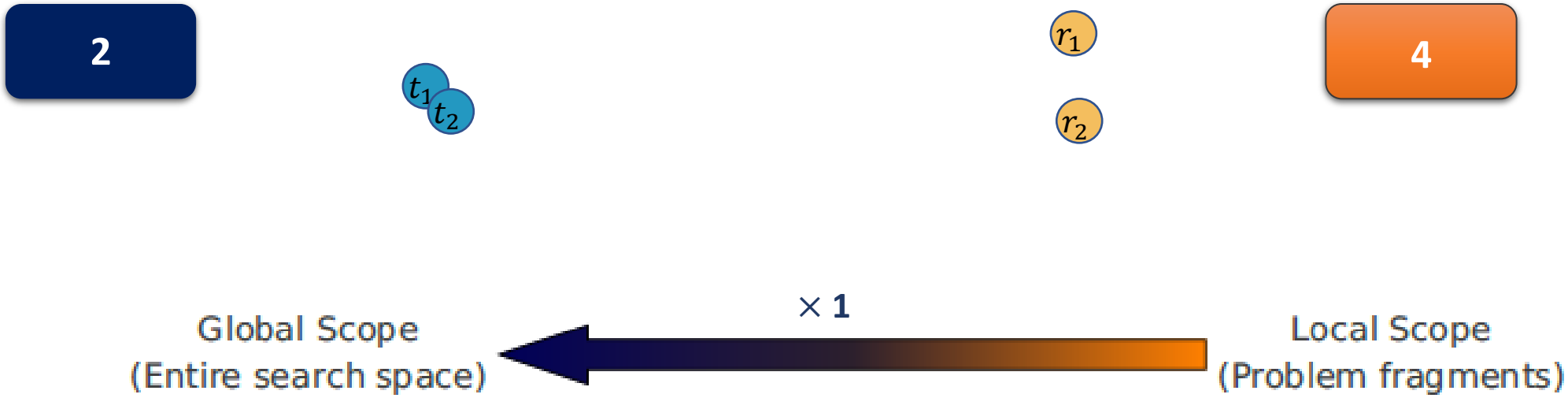


Scheduling heuristics

- Limiting the scope to problem fragments results in optimization speedup which scales with the problem size:

Number of necessary evaluations:

Mapping 2 tasks onto 2 resources.

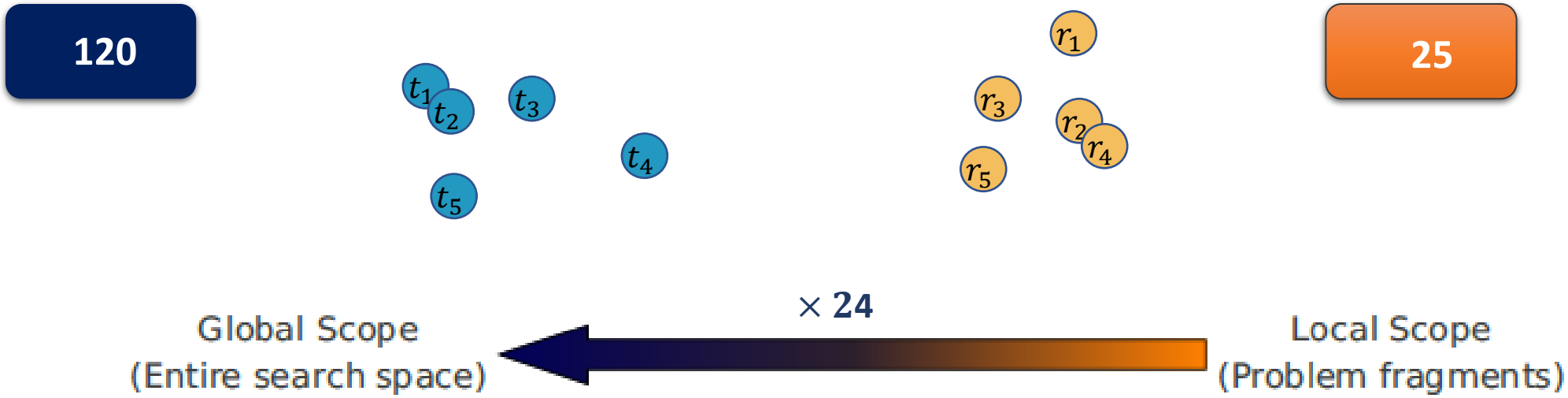


Scheduling heuristics

- Limiting the scope to problem fragments results in optimization speedup which scales with the problem size:

Number of necessary evaluations:

Mapping 5 tasks onto 5 resources.

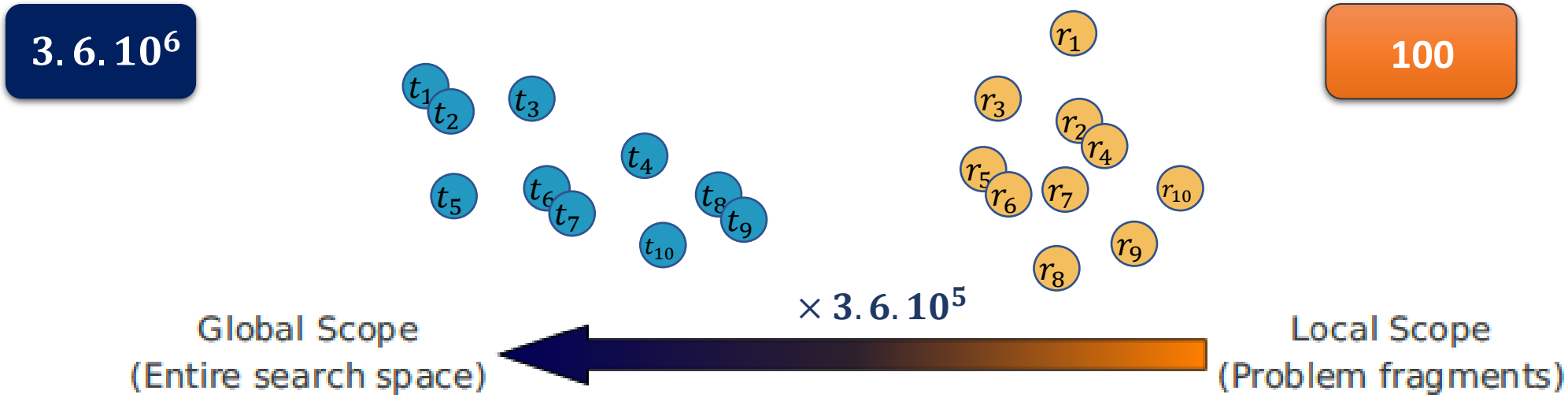


Scheduling heuristics

- Limiting the scope to problem fragments results in optimization speedup which scales with the problem size:

Number of necessary evaluations:

Mapping 10 tasks onto 10 resources.

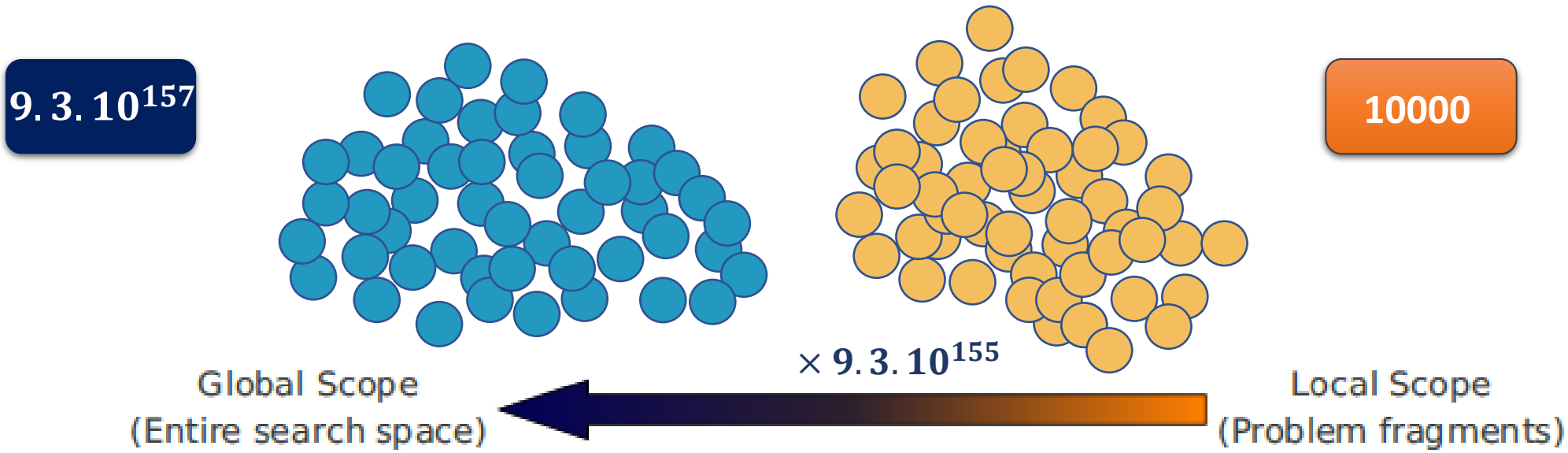


Scheduling heuristics

- Limiting the scope to problem fragments results in optimization speedup which scales with the problem size:

Number of necessary evaluations:

Mapping 100 tasks onto 100 resources.

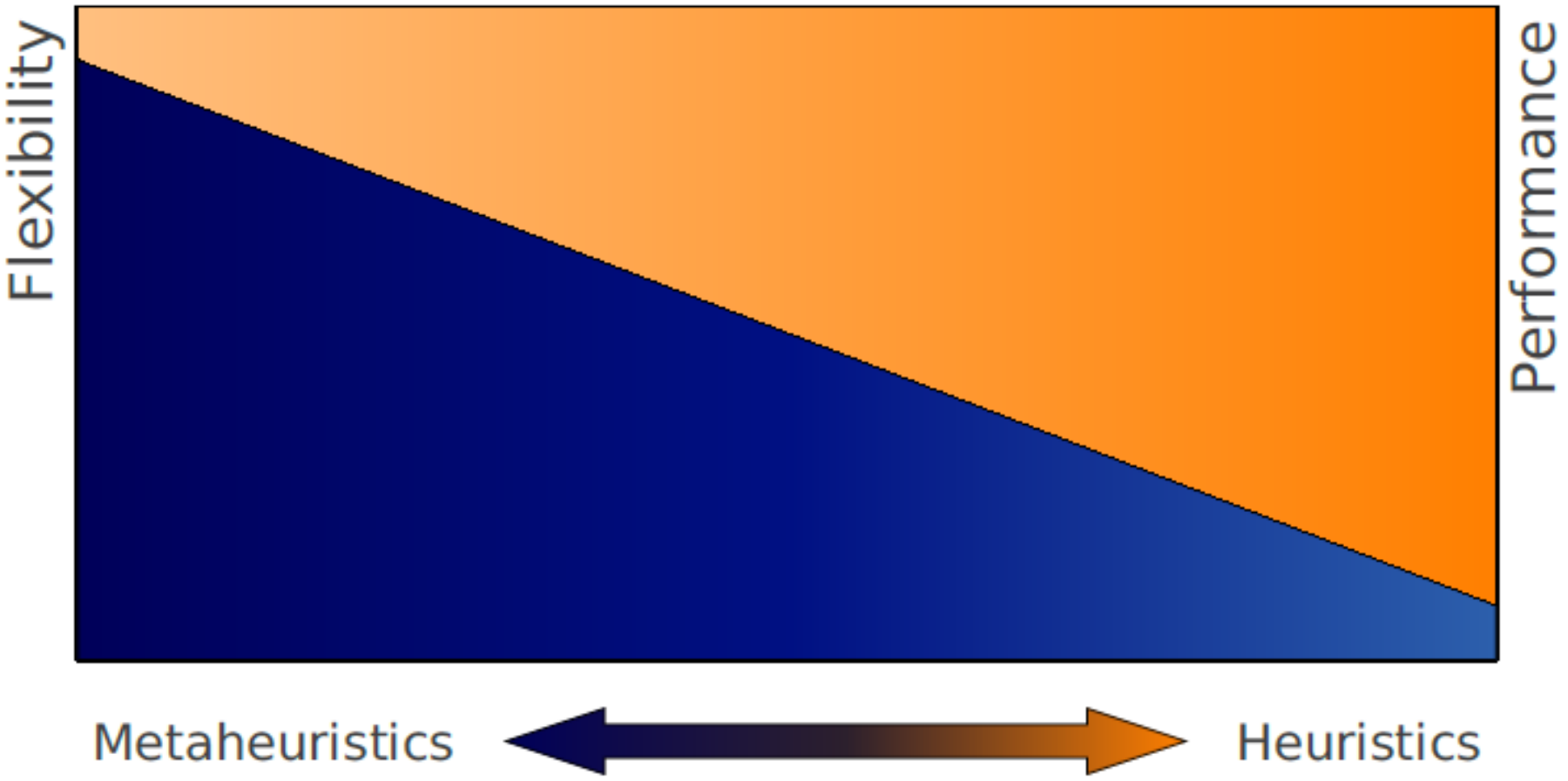


Scheduling heuristics

- Consequently, heuristic approaches...
- Achieve a high performance by **significantly reducing the search space** size (the dimensionality of the fragments is much lower than that of the whole problem).
- When making decisions for a fragment, are **blind to the implications of these decisions** for the fragments which are processed later.
- Not capable of accepting a locally suboptimal decision to obtain a globally optimal solution down the line.

Scheduling heuristics	
Advantage	Disadvantage
Very fast	Mostly (Single-objective)
Good results (for the right problem)	Problem specific

Scheduling heuristics



Scheduling heuristics in Edge-Cloud computing

Scheduling approaches in the cloud/edge domain are, as of now, mostly based on problem-specific heuristics.

This is mainly caused by...

- the large scale of the scheduling problems in this domain.
- the abstraction from many resource constraints (e.g., processing capacity) provided by the cloud providers.

Metaheuristic approaches

Algorithm types

- While the distinction between **exact** and **heuristic** algorithms is very clear, there is, within the group of heuristic algorithms, a somewhat vague distinction between **problem-specific heuristics** and **metaheuristics**.

Informal description of metaheuristic:

- "A metaheuristic is a **higher-level** procedure or heuristic designed to find, generate, or select a heuristic (partial search algorithm) that may provide **a sufficiently good solution** to an optimization problem, especially **with incomplete or imperfect information** or limited computation capacity." - Wikipedia

Heuristics vs metaheuristics approaches

Heuristics	Metaheuristics
Based on very specific assumption about the problem	Based on more generic assumption about search process
Computationally lightweight	Computationally intensive
Restricted to a certain problem	More generic

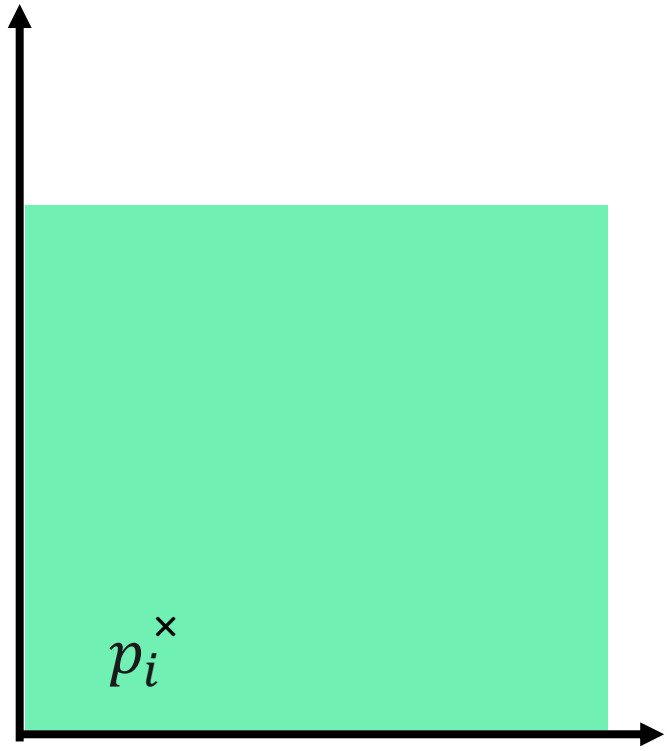
Metaheuristics meaning

Optimization approaches referred to as metaheuristics focus...

- less on the specific characteristics of the particular use case.
- Instead, their emphasis is on general characteristics that apply broadly to optimization and search processes.

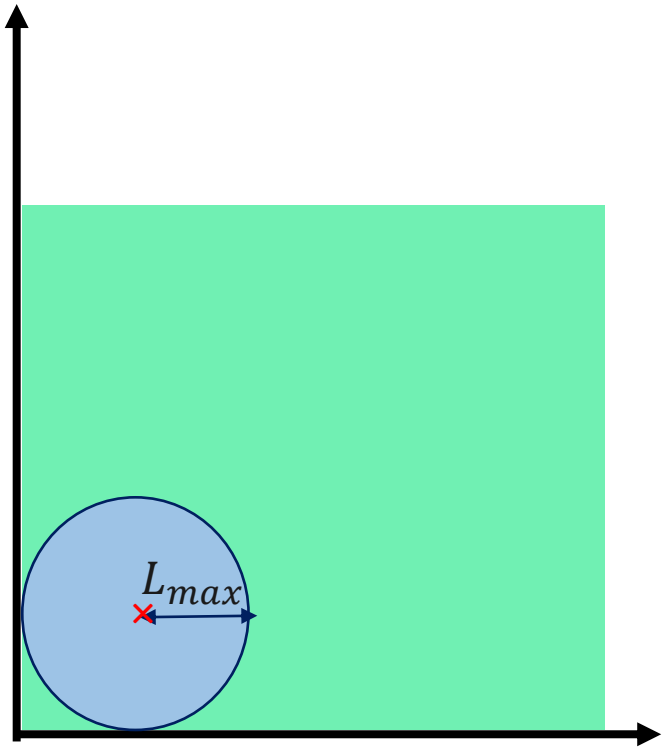
Random search in metaheuristics

- The term ***Random Search*** denotes a type of optimization approaches which operate by *iteratively* moving through the search space. In each iteration i, they relatively to the current position p_i.



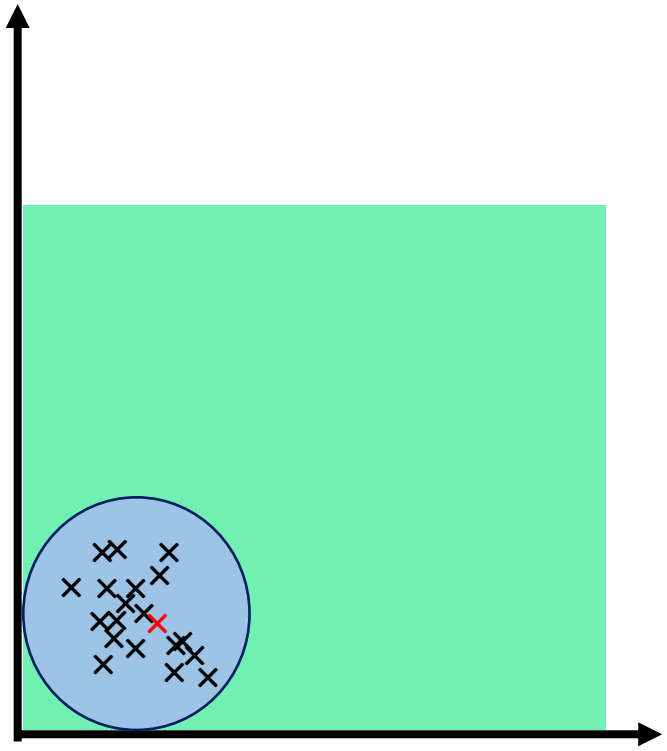
Random search in metaheuristics

- The term ***Random Search*** denotes a type of optimization approaches which operate by *iteratively* moving through the search space. In each iteration i, they relatively to the current position p_i.
- Consider the search space within a certain radius L_{max}



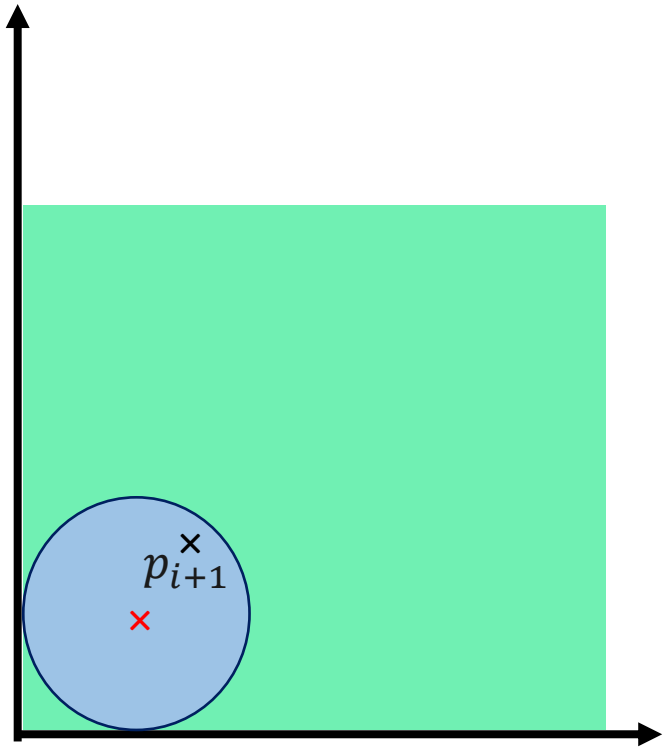
Random search in metaheuristics

- The term ***Random Search*** denotes a type of optimization approaches which operate by *iteratively* moving through the search space. In each iteration i, they relatively to the current position p_i.
- Consider the search space within a certain radius L_{max}
- From the search space they pick and evaluate a set of *n* samples



Random search in metaheuristics

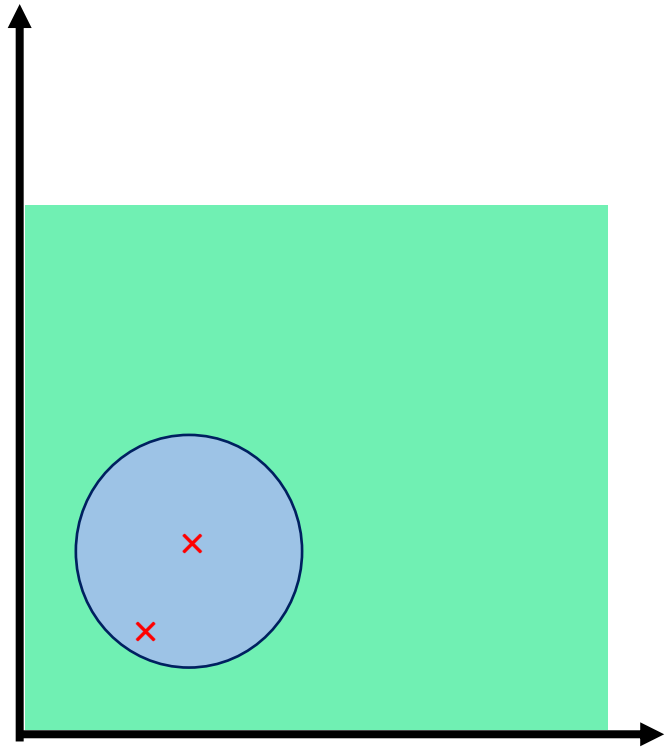
- The term ***Random Search*** denotes a type of optimization approaches which operate by *iteratively* moving through the search space. In each iteration i, they relatively to the current position p_i.
- Consider the search space within a certain radius L_{max}
- From the search space they pick and evaluate a set of *n* samples
- To use the best of the samples as p_{i+1}, the position for the next iteration.



Random search in metaheuristics

- The term ***Random Search*** denotes a type of optimization approaches which operate by *iteratively* moving through the search space. In each iteration i, they relatively to the current position p_i.

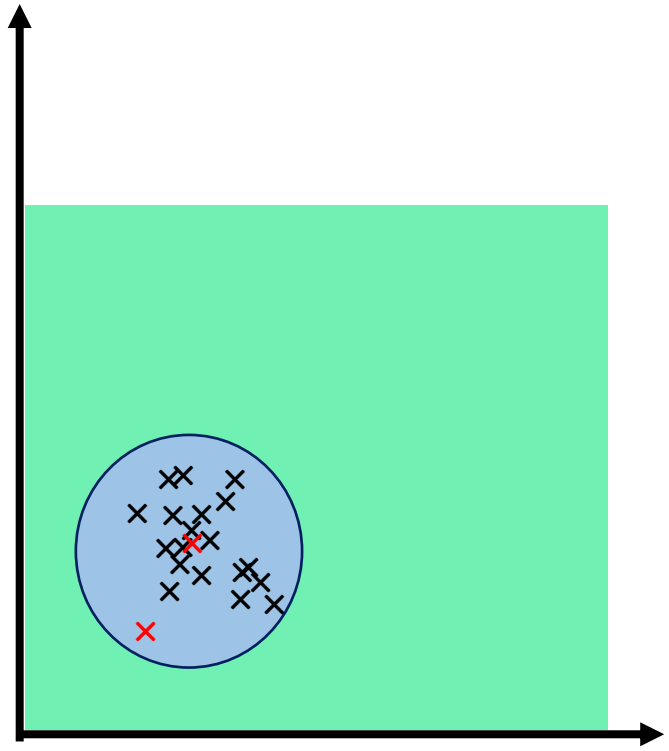
- This procedure is repeated until the algorithm reaches its *termination condition* (e.g., predefined number of iterations or lack of improvement).



Random search in metaheuristics

- The term ***Random Search*** denotes a type of optimization approaches which operate by *iteratively* moving through the search space. In each iteration i, they relatively to the current position p_i.

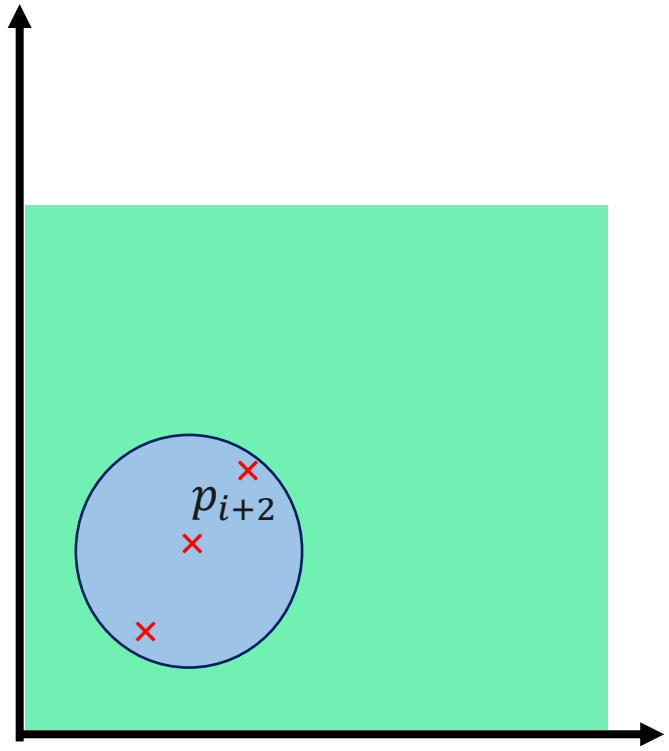
- This procedure is repeated until the algorithm reaches its *termination condition* (e.g., predefined number of iterations or lack of improvement).



Random search in metaheuristics

- The term ***Random Search*** denotes a type of optimization approaches which operate by *iteratively* moving through the search space. In each iteration i, they relatively to the current position p_i.

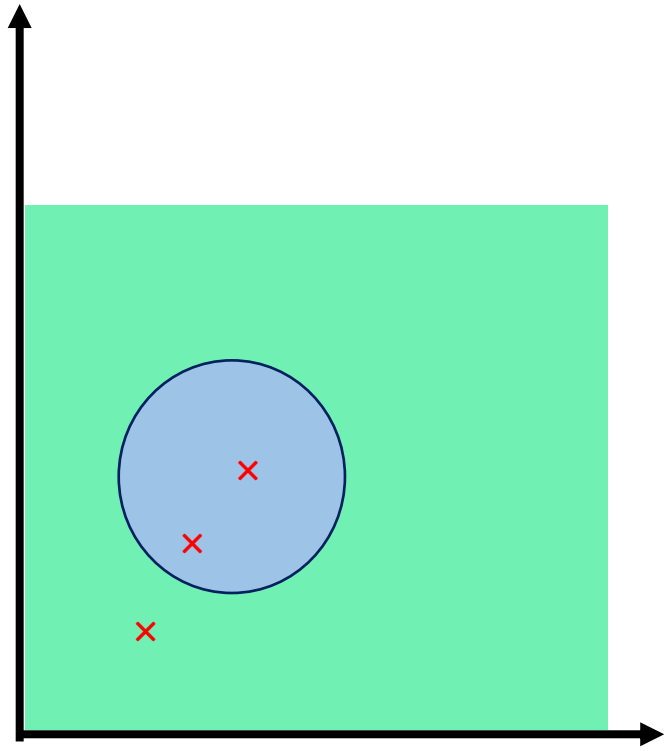
- This procedure is repeated until the algorithm reaches its *termination condition* (e.g., predefined number of iterations or lack of improvement).



Random search in metaheuristics

- The term ***Random Search*** denotes a type of optimization approaches which operate by *iteratively* moving through the search space. In each iteration i, they relatively to the current position p_i.

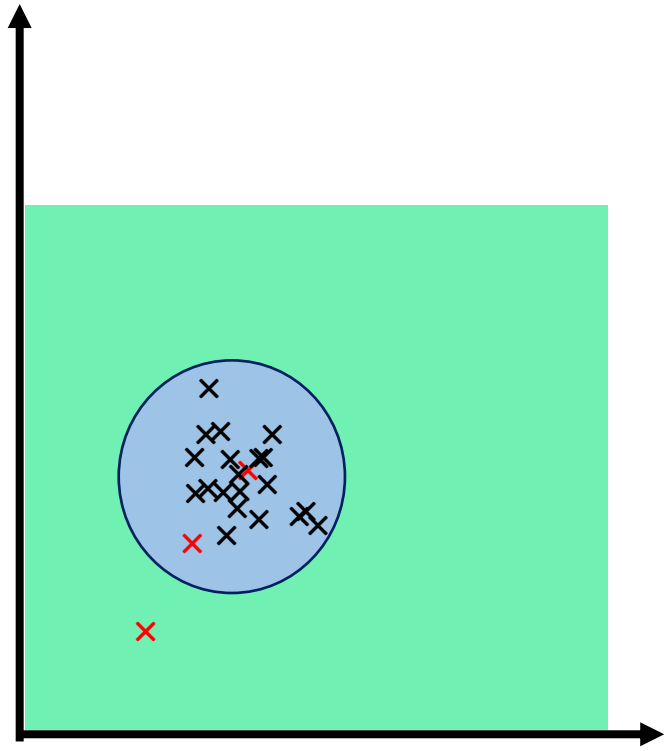
- This procedure is repeated until the algorithm reaches its *termination condition* (e.g., predefined number of iterations or lack of improvement).



Random search in metaheuristics

- The term ***Random Search*** denotes a type of optimization approaches which operate by *iteratively* moving through the search space. In each iteration i, they relatively to the current position p_i.

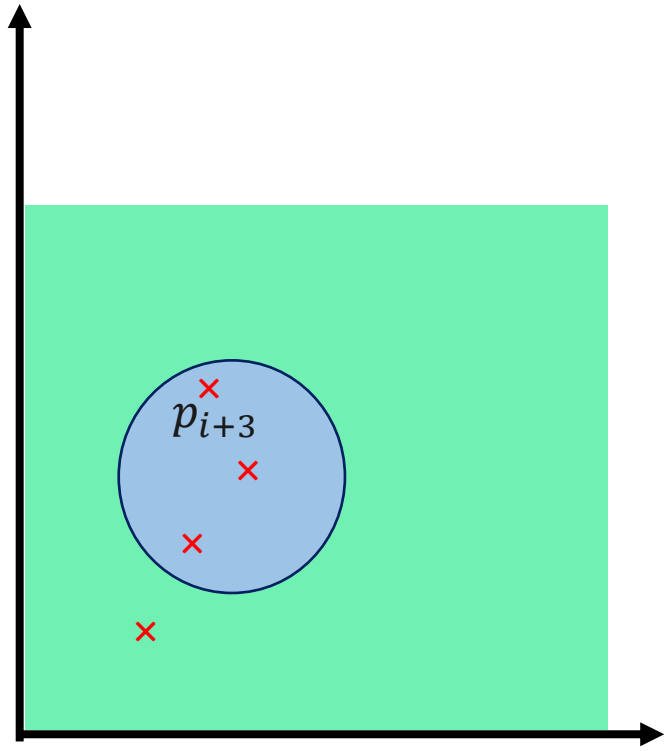
- This procedure is repeated until the algorithm reaches its *termination condition* (e.g., predefined number of iterations or lack of improvement).



Random search in metaheuristics

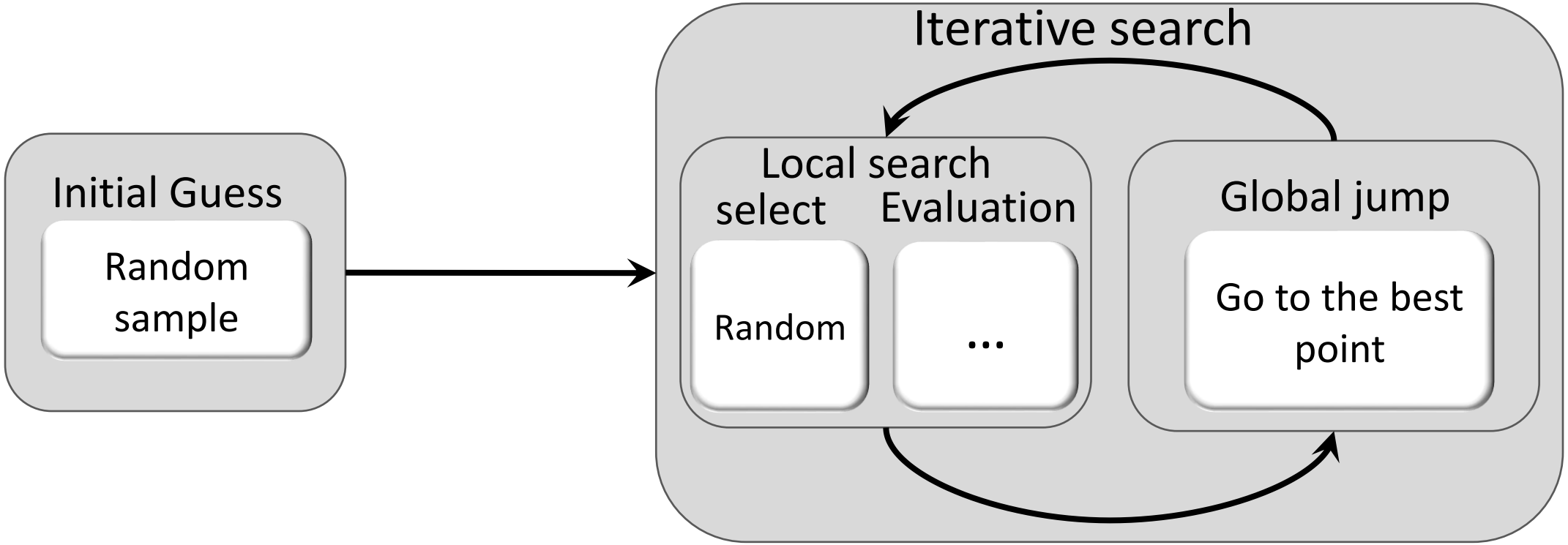
- The term ***Random Search*** denotes a type of optimization approaches which operate by *iteratively* moving through the search space. In each iteration i, they relatively to the current position p_i.

- This procedure is repeated until the algorithm reaches its *termination condition* (e.g., predefined number of iterations or lack of improvement).



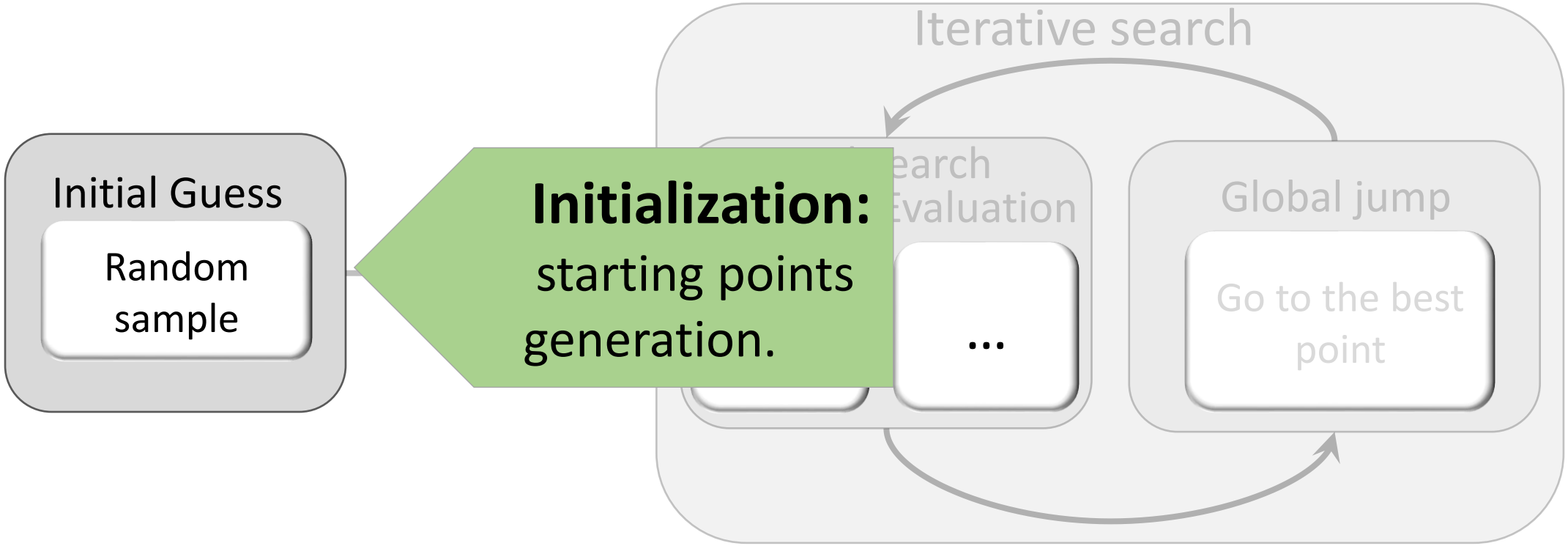
General procedure in metaheuristics

- Due to its simplicity, the random search algorithm is particularly well suited to demonstrate the general procedure of iterative metaheuristics:



General procedure in metaheuristics

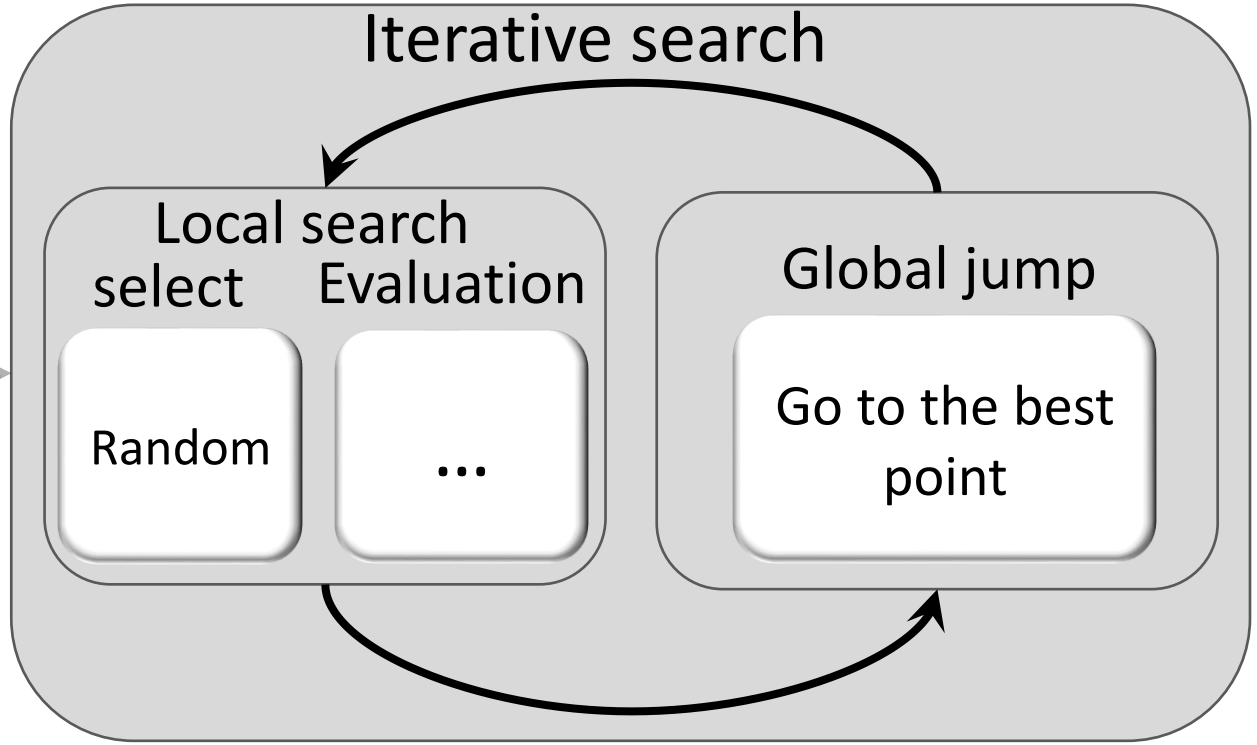
- Due to its simplicity, the random search algorithm is particularly well suited to demonstrate the general procedure of iterative metaheuristics:



General procedure in metaheuristics

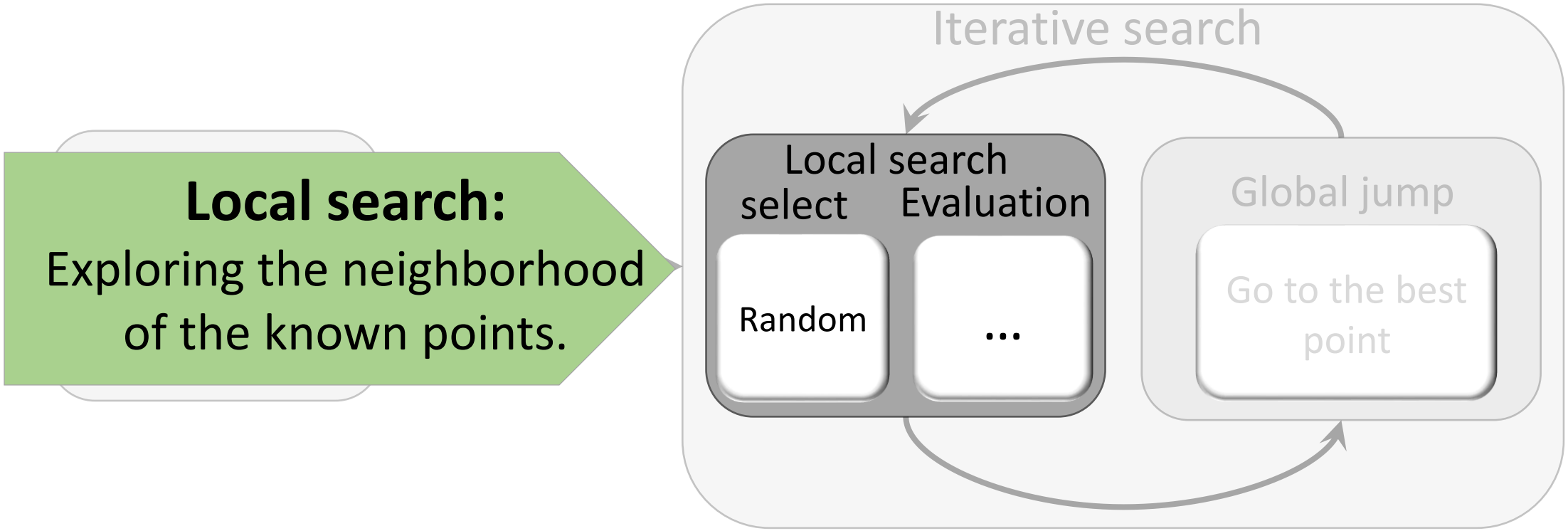
- Due to its simplicity, the random search algorithm is particularly well suited to demonstrate the general procedure of iterative metaheuristics:

Optimization Loop:
Iterative search space exploration.



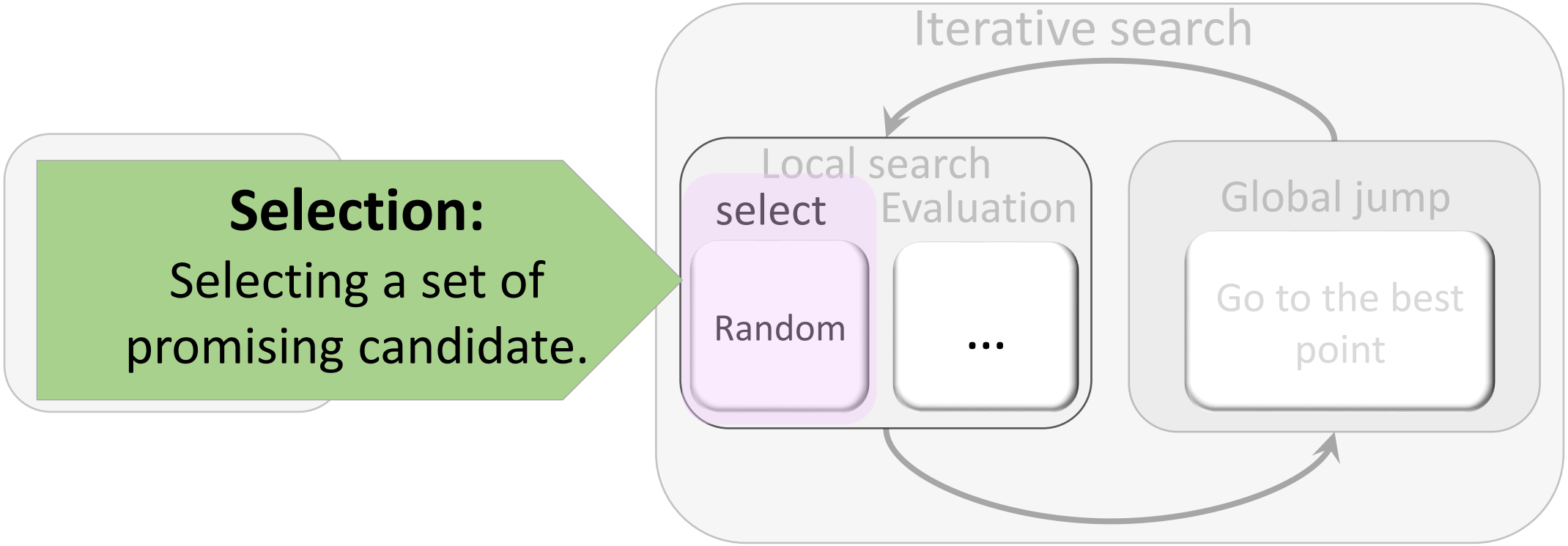
General procedure in metaheuristics

- Due to its simplicity, the random search algorithm is particularly well suited to demonstrate the general procedure of iterative metaheuristics:



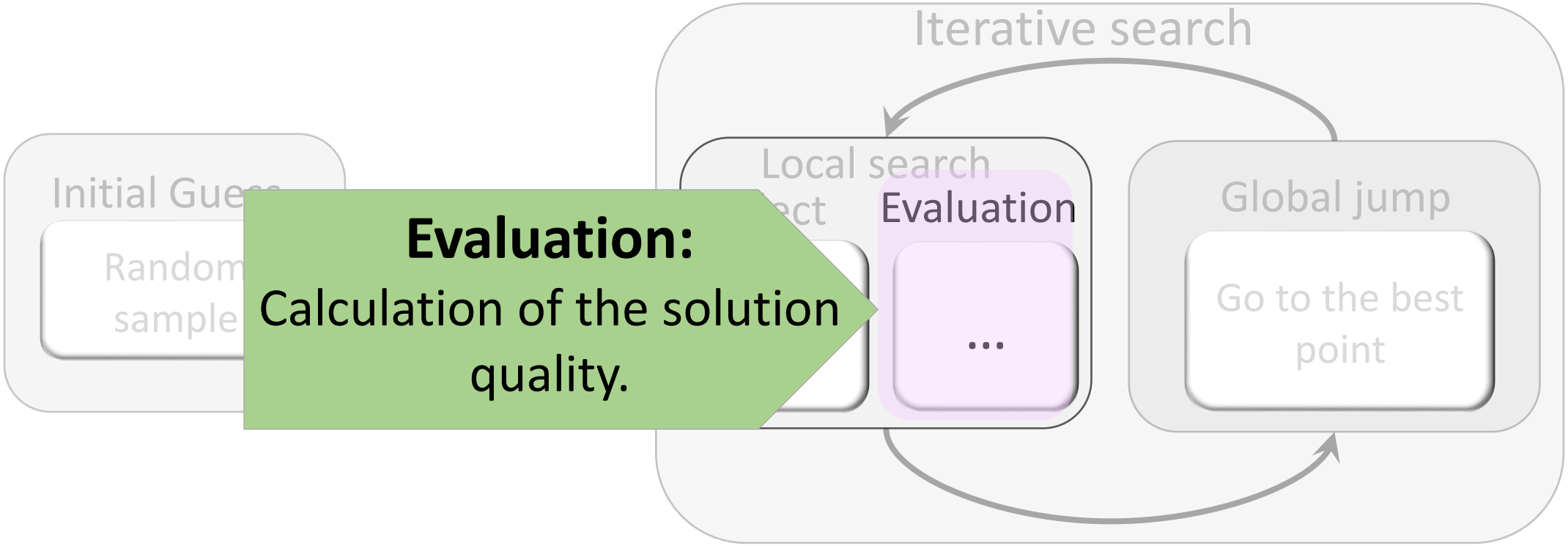
General procedure in metaheuristics

- Due to its simplicity, the random search algorithm is particularly well suited to demonstrate the general procedure of iterative metaheuristics:



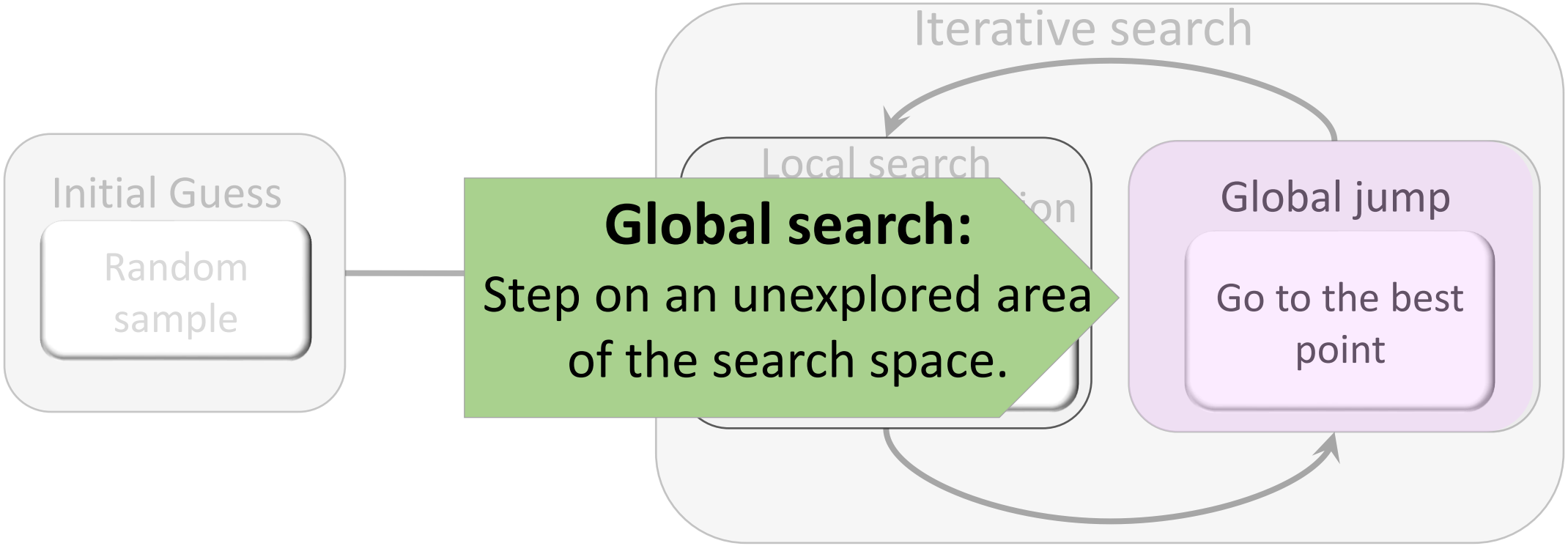
General procedure in metaheuristics

- Due to its simplicity, the random search algorithm is particularly well suited to demonstrate the general procedure of iterative metaheuristics:



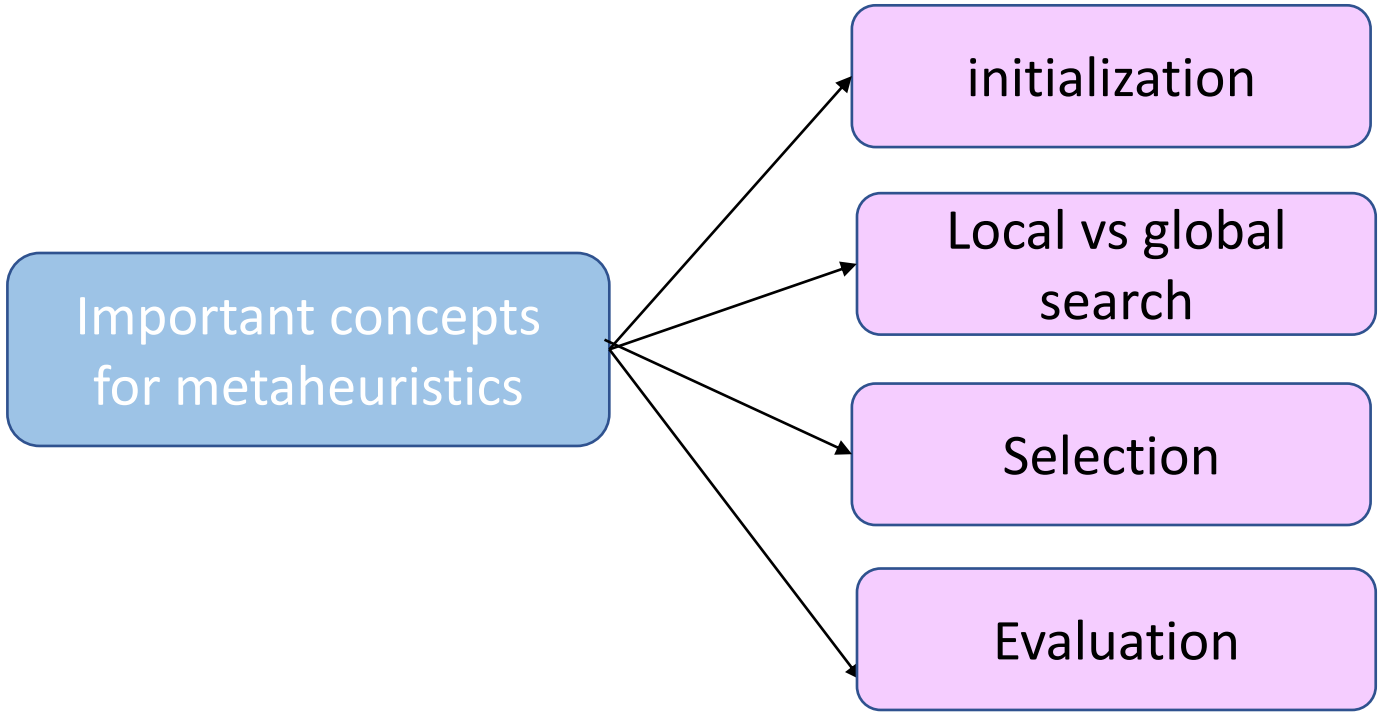
General procedure in metaheuristics

- Due to its simplicity, the random search algorithm is particularly well suited to demonstrate the general procedure of iterative metaheuristics:



General procedure in metaheuristics

- While most metaheuristics follow the general procedure, their individual differences result from different implementations of the individual steps.



Specific knowledge in metaheuristics

- Metaheuristic approaches use problem-specific knowledge throughout the optimization.
- The usage of evaluation functions, which are defined by the problem at hand, obviously introduces problem-specific knowledge into the optimization.
- During the local search, metaheuristics explore the *neighborhood* of the current position.

Neighborhood in metaheuristics

- Informal description:

The neighborhood of a point within the search space is given by the collection of all points which are located in the vicinity of this point.

- Formal definition:

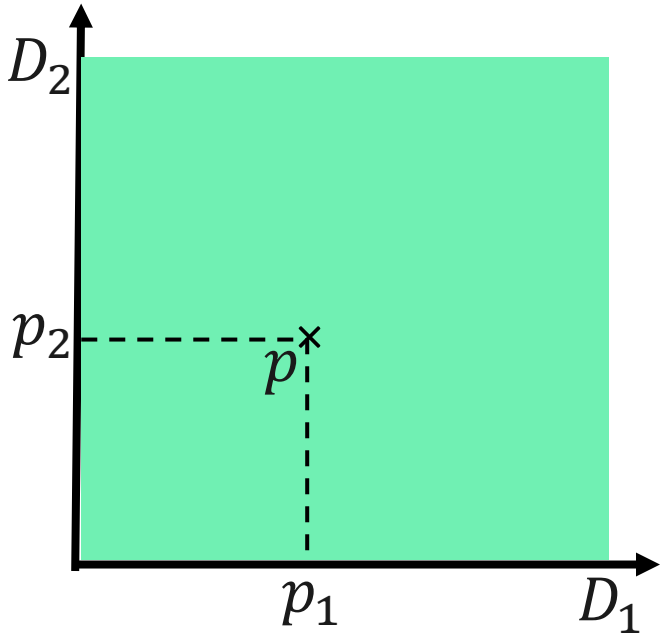
For a given search space point $\boldsymbol{p}$, a point $\widetilde{\boldsymbol{p}}$ is considered as a *neighbor* of $\boldsymbol{p}$ if the *distance* (given by a problem-specific similarity metric) between $\boldsymbol{p}$ and $\widetilde{\boldsymbol{p}}$ is smaller than a predefined threshold value $\boldsymbol{L_{max}}$:

$$\widetilde{\boldsymbol{p}} \, \boldsymbol{\epsilon} \, \boldsymbol{N}(\boldsymbol{p}) \Leftrightarrow \|\boldsymbol{p} - \widetilde{\boldsymbol{p}}\| < \boldsymbol{L_{max}}$$

where $\boldsymbol{N}(\boldsymbol{p})$ denotes the *neighborhood* of $\boldsymbol{p}$, i.e., the set of all neighbors of $\boldsymbol{p}$.

Problem specific knowledge in metaheuristics

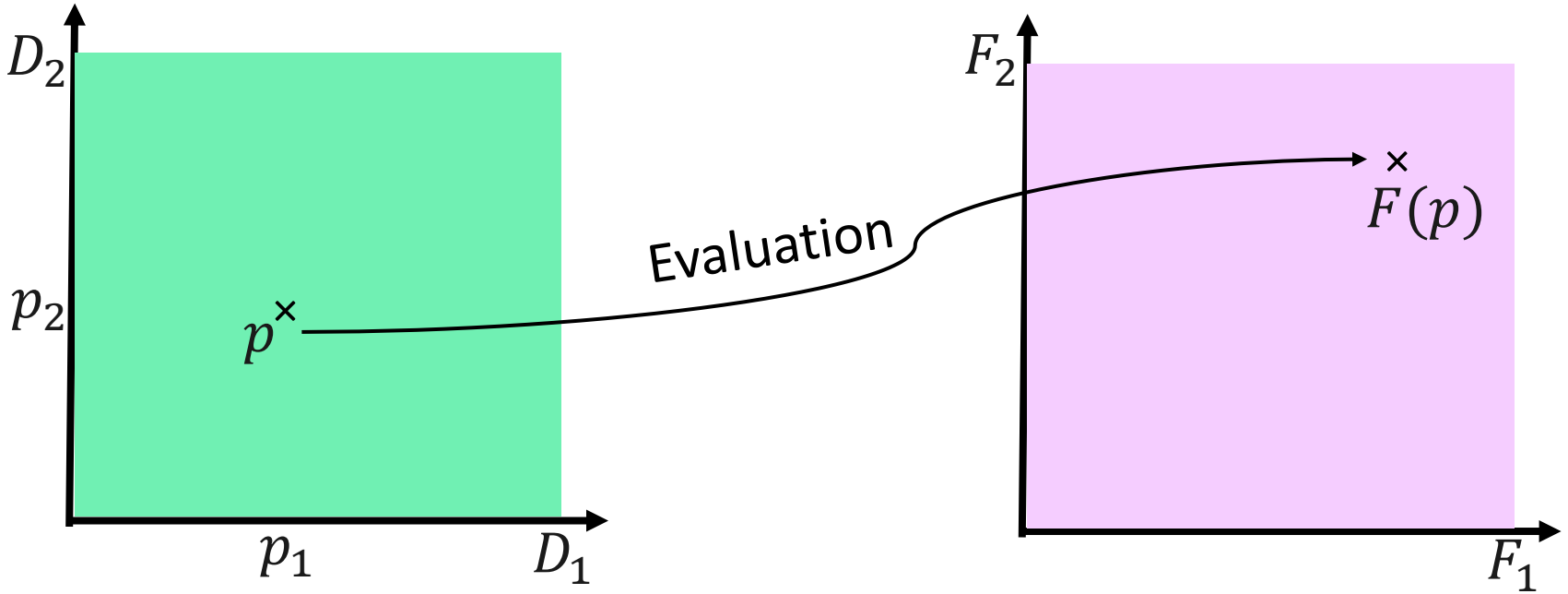
- A fundamental assumption (implicitly) made about optimization problems is a ***similarity assumption*** between the search and the *objective* space.



Search space

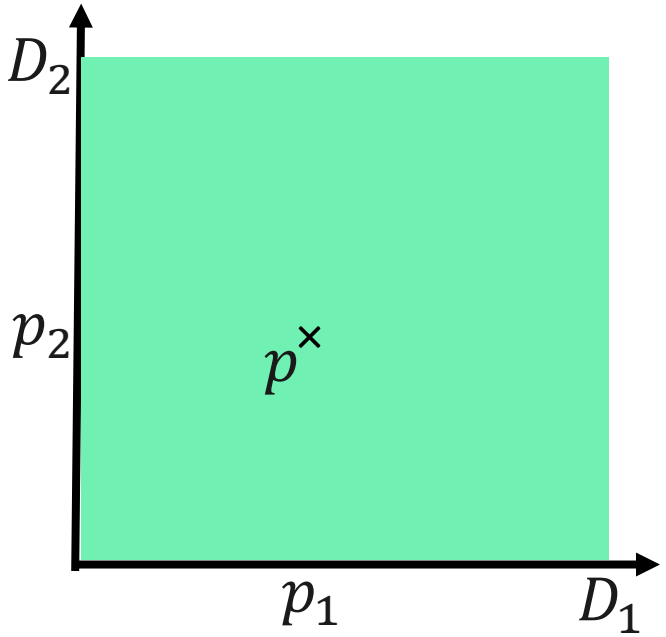
Problem specific knowledge in metaheuristics

- A fundamental assumption (implicitly) made about optimization problems is a ***similarity assumption*** between the search and the *objective* space.

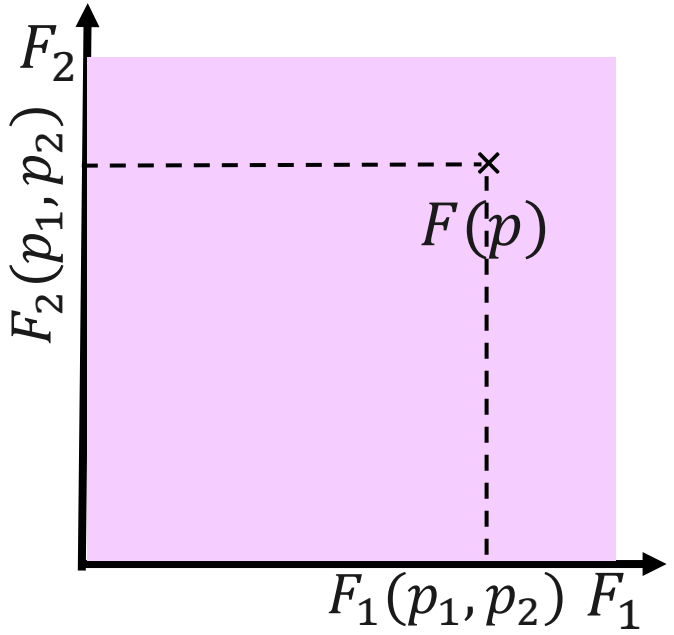


Problem specific knowledge in metaheuristics

- A fundamental assumption (implicitly) made about optimization problems is a ***similarity assumption*** between the search and the *objective* space.



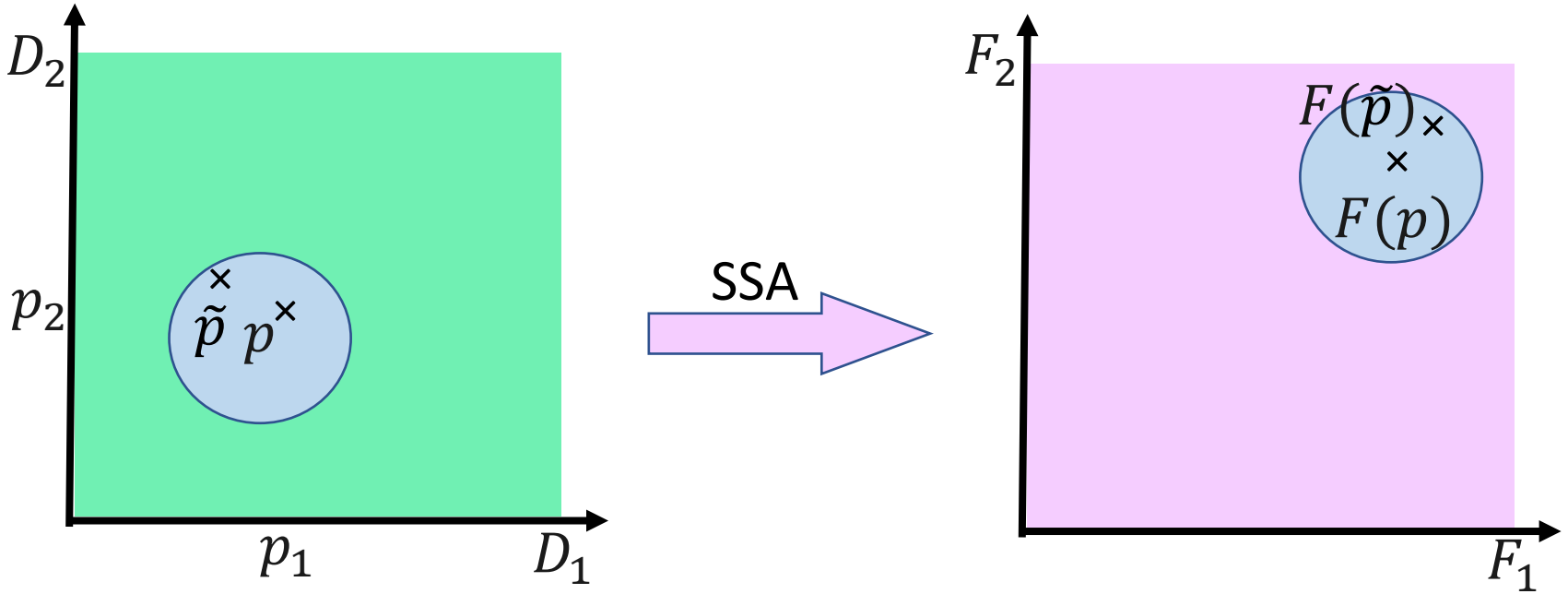
Search space



Objective space

Problem specific knowledge in metaheuristics

- A fundamental assumption (implicitly) made about optimization problems is a ***similarity assumption*** between the search and the *objective* space.



Search space

Objective space

Spatial similarity assumption (SSA)

- Informal description:

Assumption that search space points created by a similar set of design decisions have a similar quality considering the design objectives.

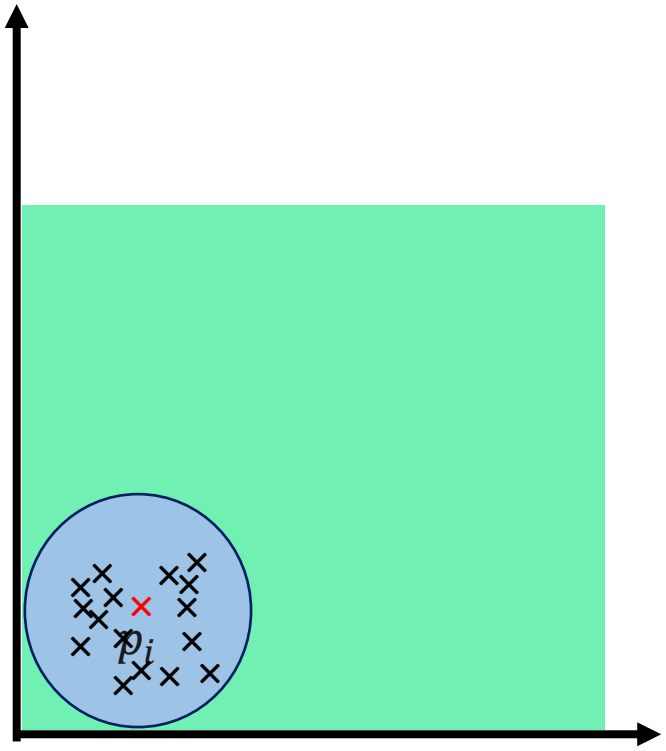
- Formal definition:

If point $\boldsymbol{p}$ and point $\widetilde{\boldsymbol{p}}$ are neighbors, their *objective projections* are neighbors in the objective space:

$$\boldsymbol{p} \in N(\widetilde{\boldsymbol{p}}) \wedge \widetilde{\boldsymbol{p}} \in N(\boldsymbol{p}) \Rightarrow F(\boldsymbol{p}) \in N(F(\widetilde{\boldsymbol{p}})) \wedge F(\widetilde{\boldsymbol{p}}) \in N(F(\boldsymbol{p}))$$

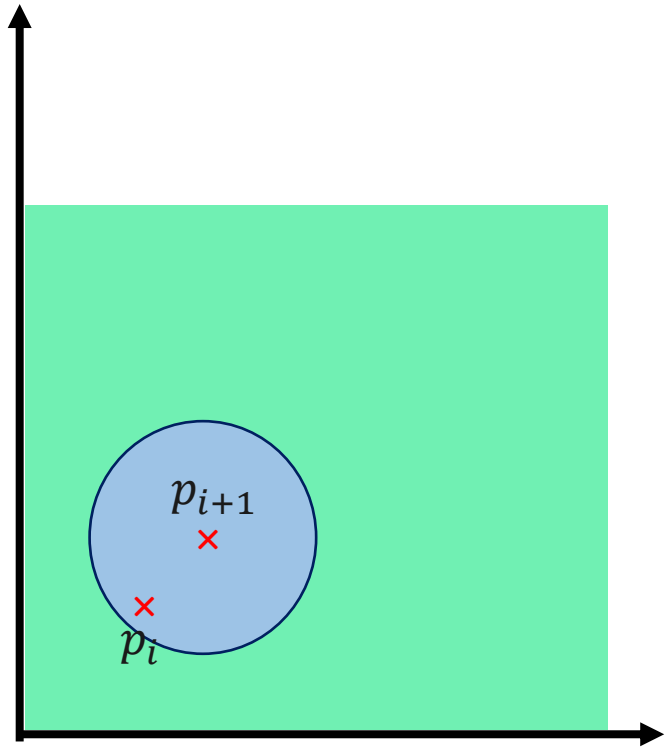
Spatial similarity assumption (SSA)

- Only because of the SSA, it is reasonable to...
- estimate the characteristics of an entire area of the search space after examining a set of a few samples...



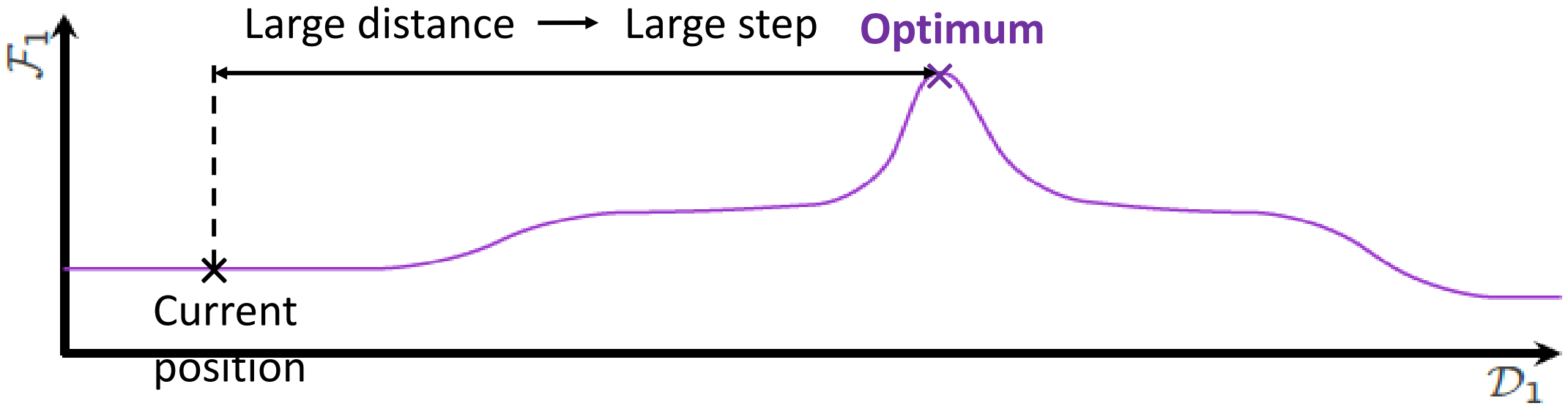
Spatial similarity assumption (SSA)

- Only because of the SSA, it is reasonable to...
- estimate the characteristics of an entire area of the search space after examining a set of a few samples...
- identify a promising area for the next local search after examining a single point from it.



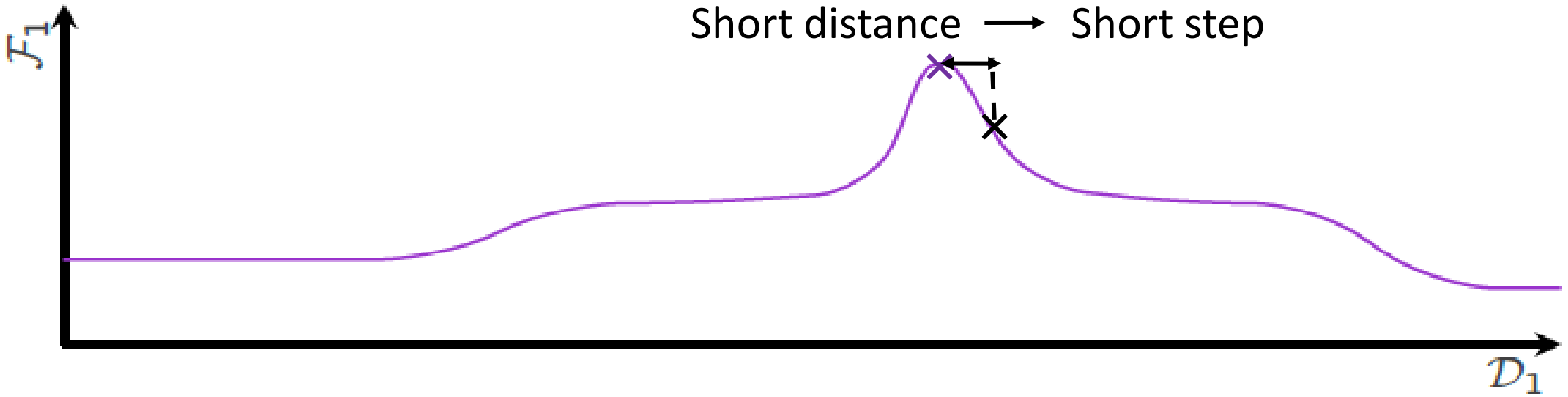
Spatial similarity assumption (SSA)

For an efficient optimization, it is important that the optimizer can adjust the width of the steps which it takes within the objective space:



Spatial similarity assumption (SSA)

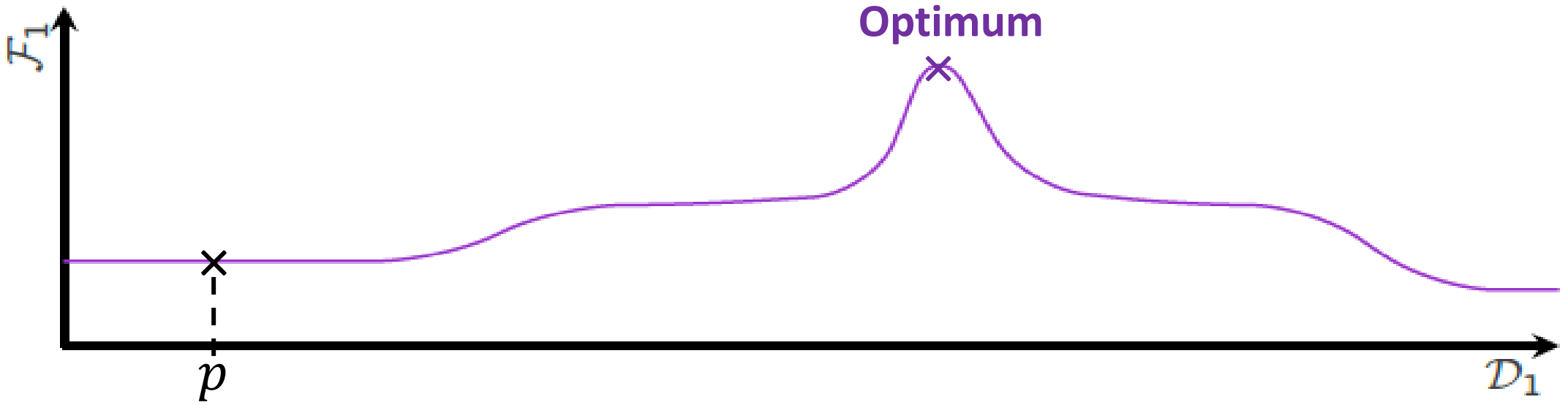
For an efficient optimization, it is important that the optimizer can adjust the width of the steps which it takes within the objective space:



Spatial similarity assumption (SSA)

In terms of spatial distances, the SSA implies that:

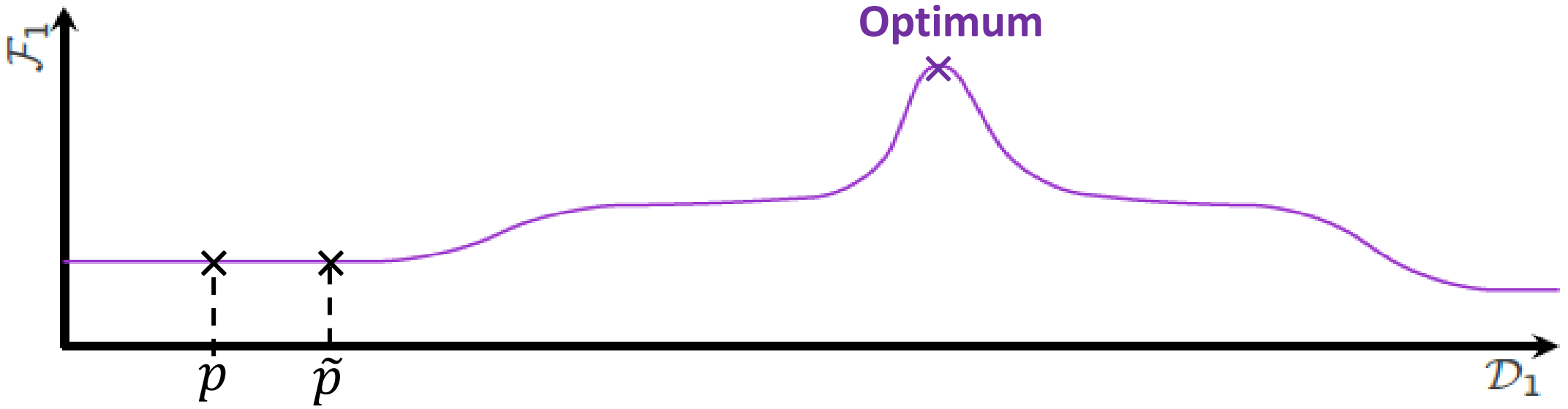
- Points which are **close** to each other in search space are **close** to each other in the objective space.



Spatial similarity assumption (SSA)

In terms of spatial distances, the SSA implies that:

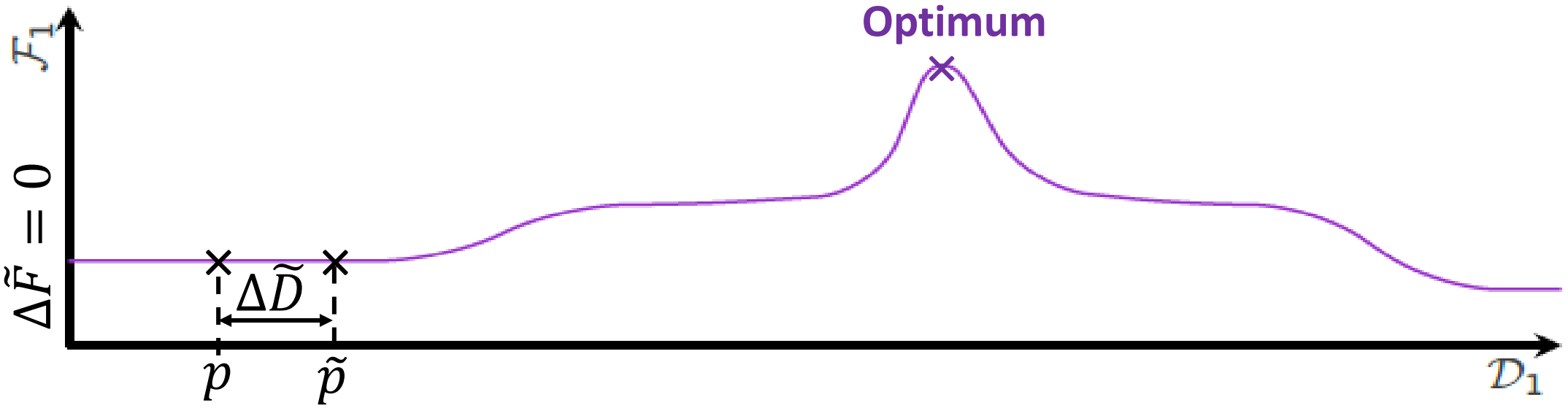
- Points which are **close** to each other in search space are **close** to each other in the objective space.



Spatial similarity assumption (SSA)

In terms of spatial distances, the SSA implies that:

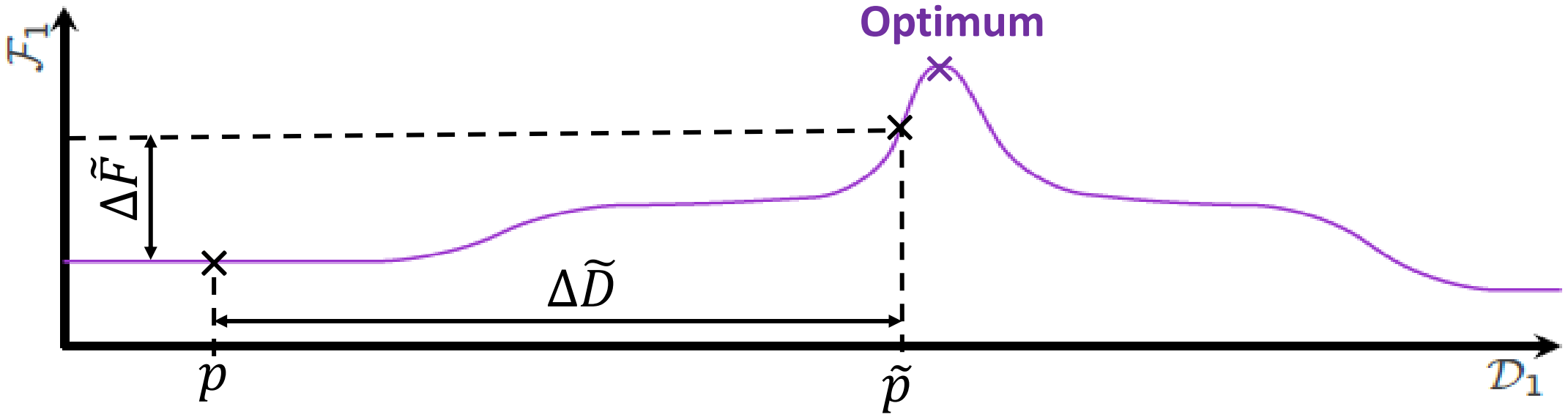
- Points which are **close** to each other in search space are **close** to each other in the objective space.



Spatial similarity assumption (SSA)

In terms of spatial distances, the SSA implies that:

- Points which are **close** to each other in search space are **close** to each other in the objective space.
- Points which are **far away** from each other in the search space are also **far away** from each other in the objective space.

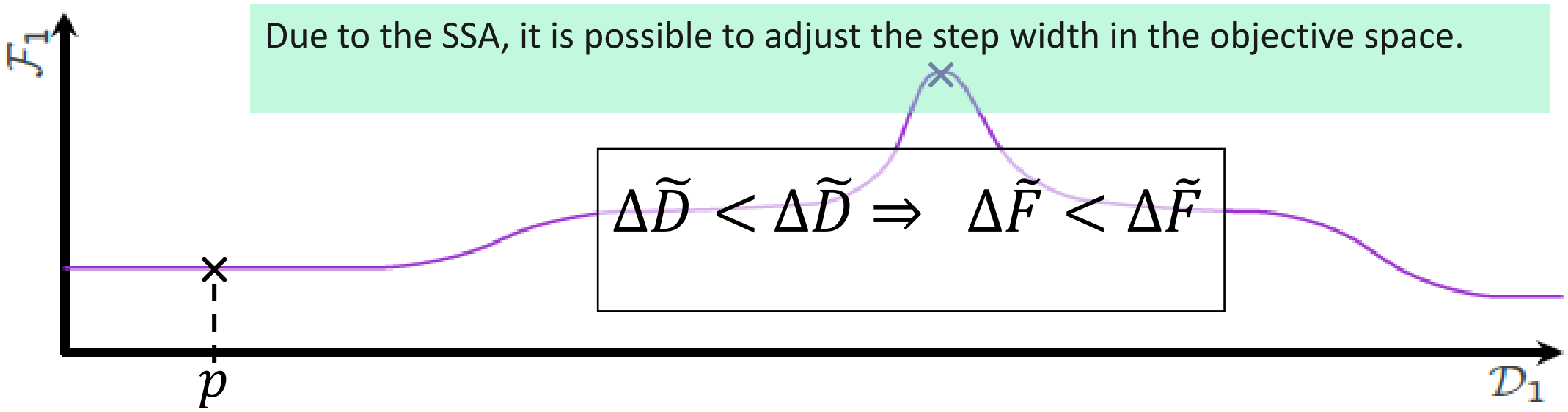


Spatial similarity assumption (SSA)

In terms of spatial distances, the SSA implies that:

- Points which are **close** to each other in search space are **close** to each other in the objective space.
- Points which are **far away** from each other in the search space are also **far away** from each other in the objective space.

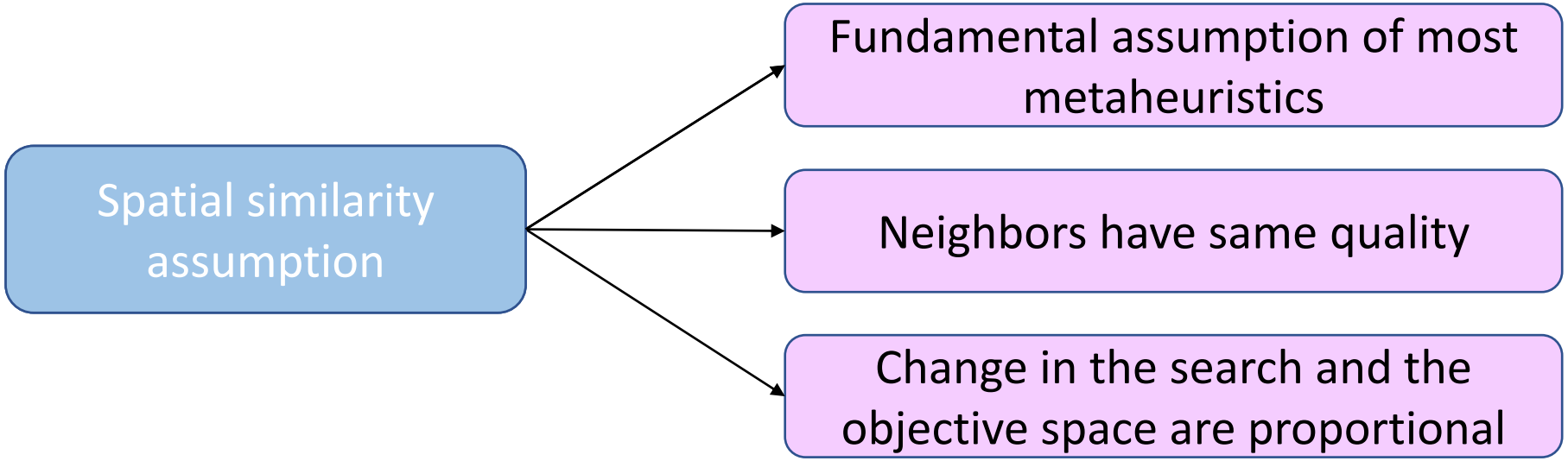
Due to the SSA, it is possible to adjust the step width in the objective space.



Spatial similarity assumption (SSA)

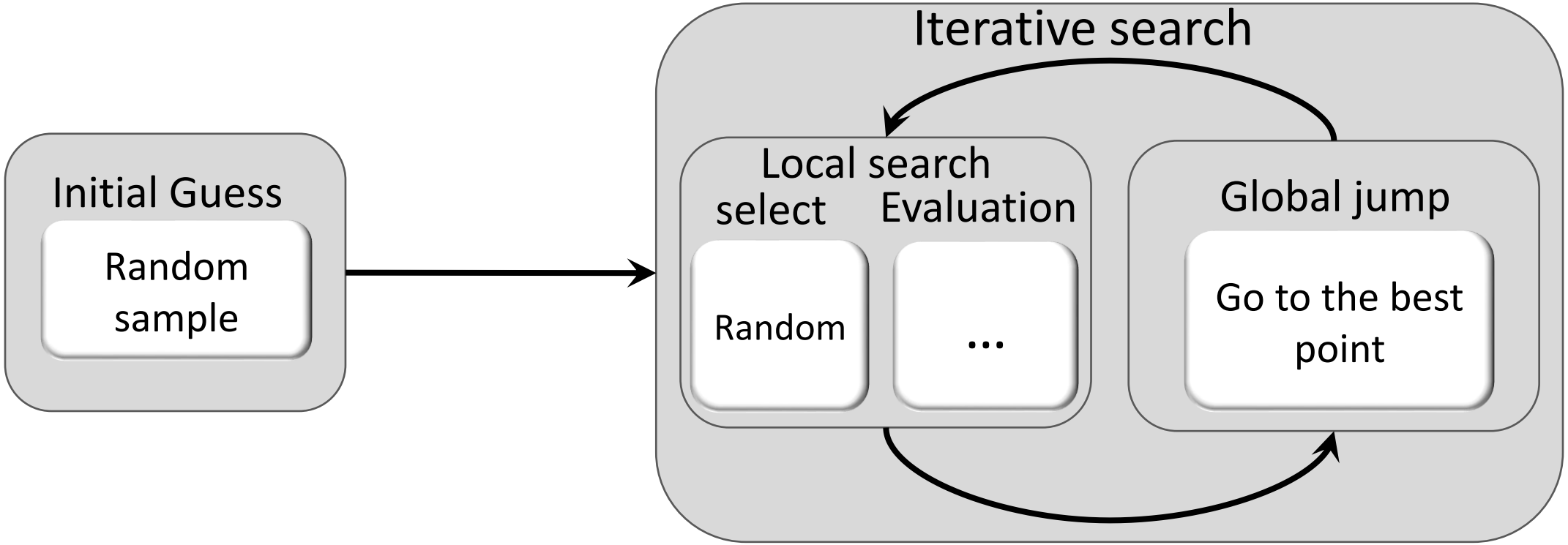
The SSA is...

- to some extent, implicitly made by most optimization metaheuristics.
- naturally fulfilled by many real-world problems.
- something that many users of optimization approaches are not aware of.



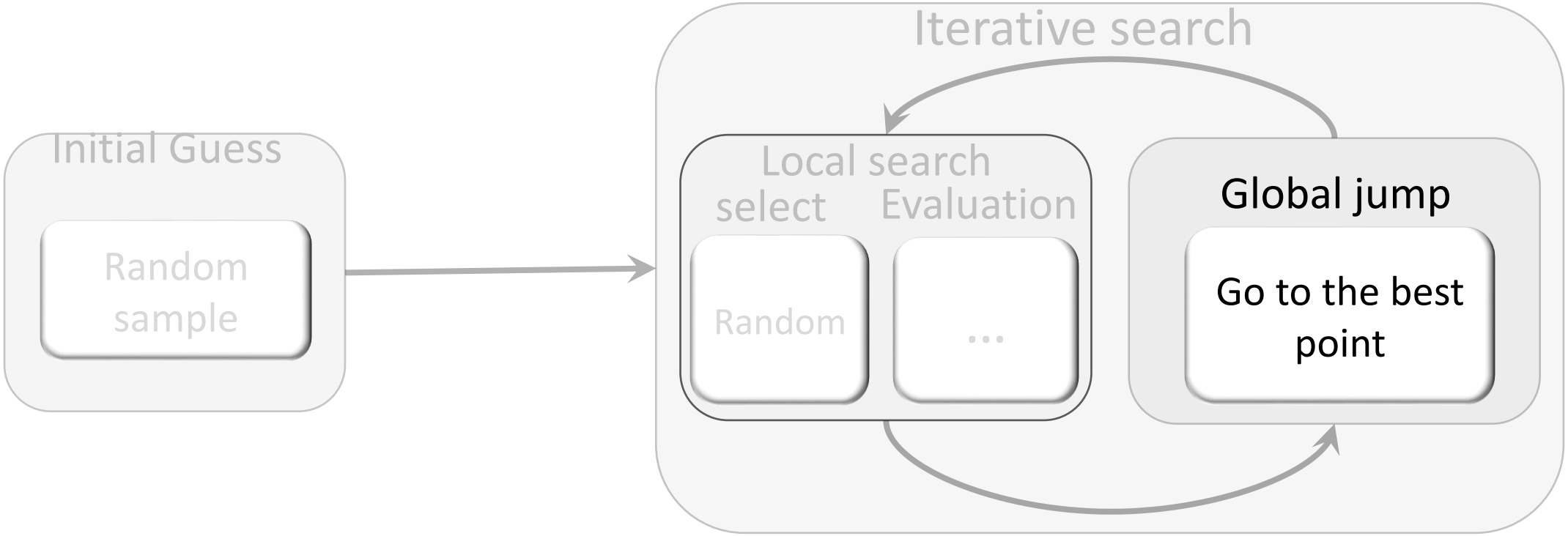
Problem specific knowledge

- With the SSA, we make a strong assumption about the problem specific knowledge (which, however, is naturally fulfilled for a large portion of real problems).



X/X Ratio in metaheuristics

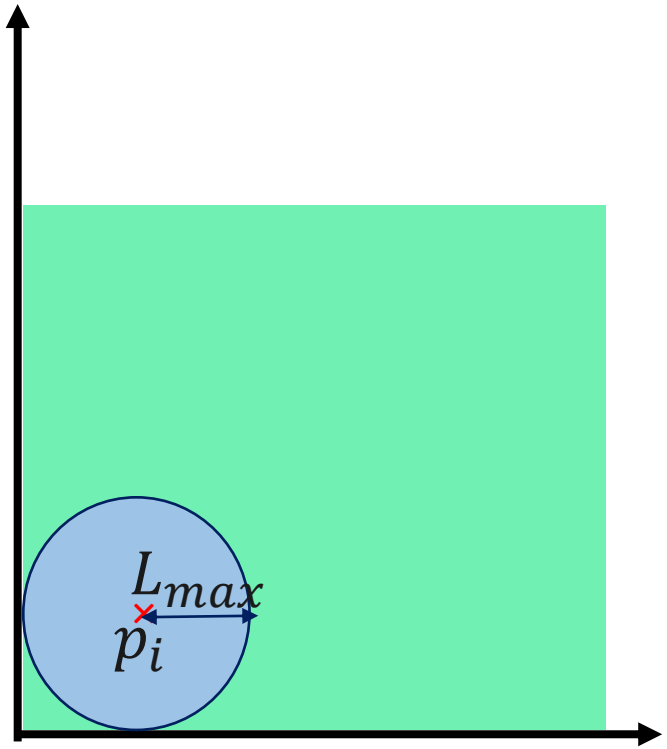
Many (but not all) optimization metaheuristics are built on a distinction between a *local* (more thorough search of a small area, smaller steps through the space) and a *global* (coarse-grained search, large steps) search.



X/X Ratio in metaheuristics

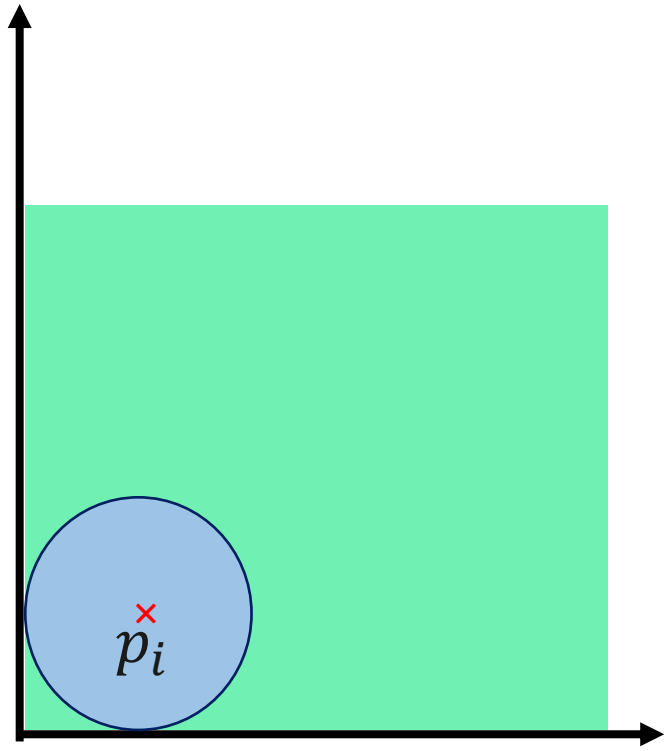
- In the random search algorithm, the local search corresponds to the exploration of the neighborhood, while the global search/jump corresponds to the step to the best currently known solution.

- **Q: How does the radius L_{max} affect the optimization?**



X/X Ratio in metaheuristics

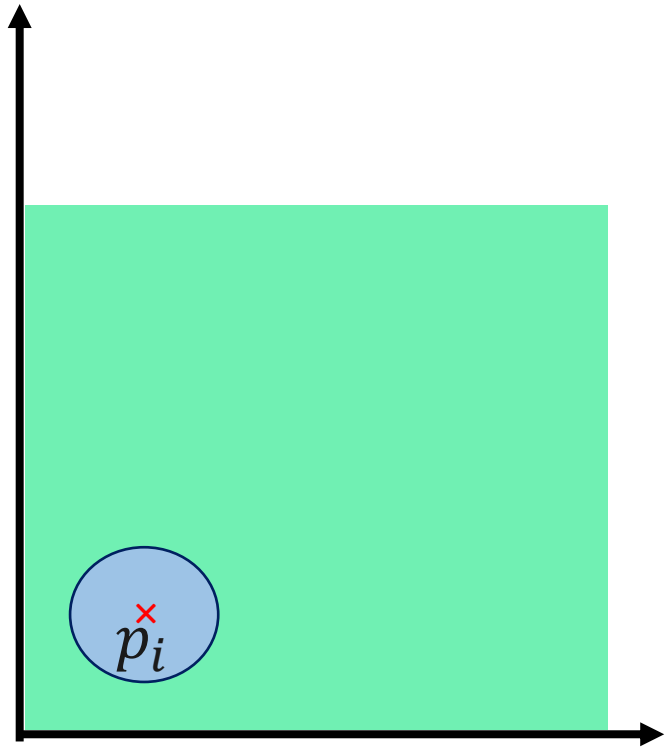
Assuming a constant number *n* of samples evaluated in each neighborhood, a **smaller** search radius L_{max} results in...



X/X Ratio in metaheuristics

Assuming a constant number *n* of samples evaluated in each neighborhood, a **smaller** search radius L_{max} results in...

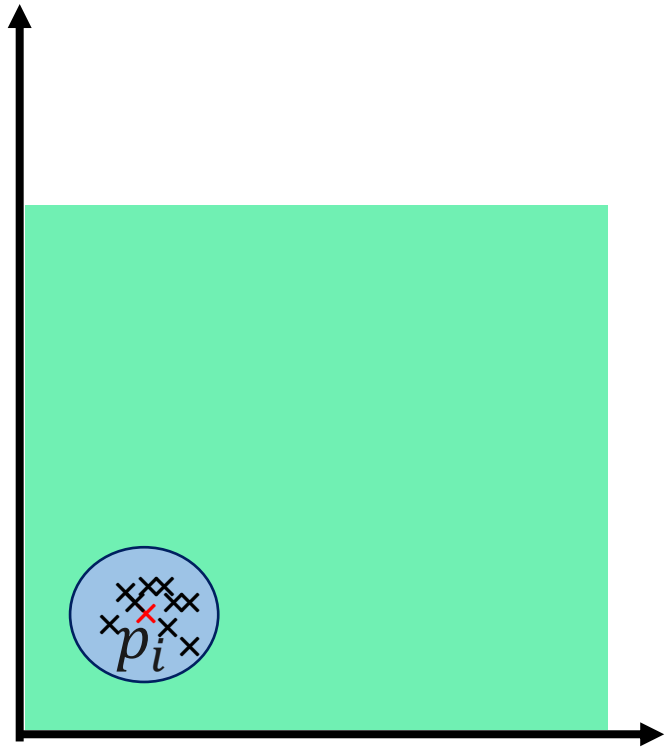
- a **more thorough** search of the neighborhood.



X/X Ratio in metaheuristics

Assuming a constant number *n* of samples evaluated in each neighborhood, a **smaller** search radius L_{max} results in...

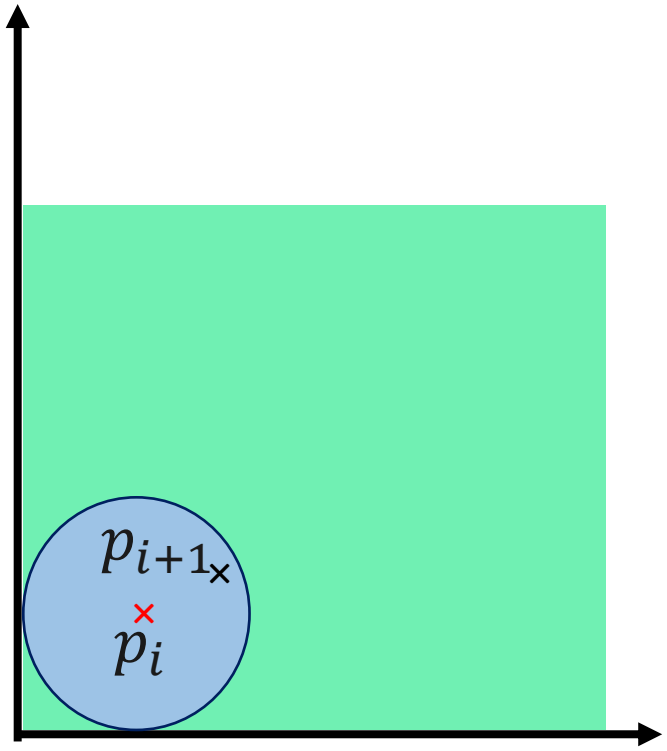
- a **more thorough** search of the neighborhood.
- a **more precise** estimation of the neighborhood quality.



X/X Ratio in metaheuristics

Assuming a constant number *n* of samples evaluated in each neighborhood, a **smaller** search radius L_{max} results in...

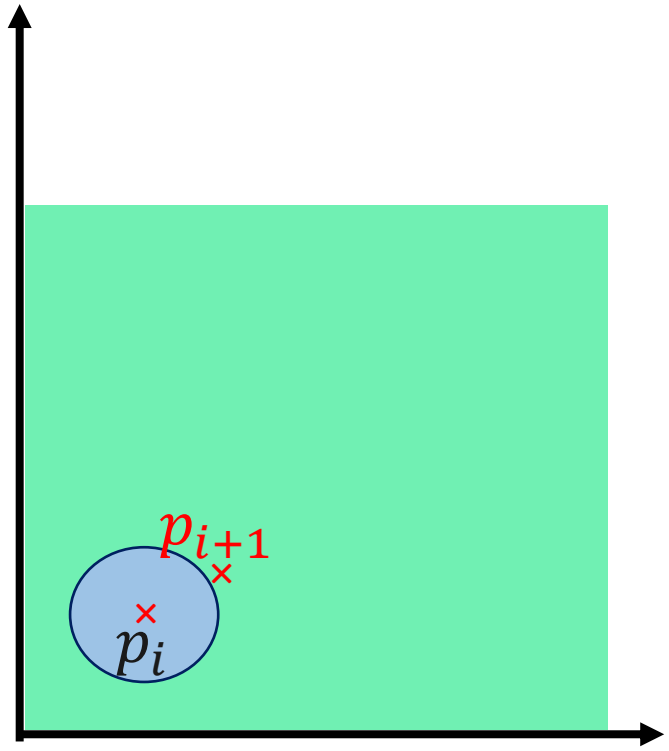
- a **more thorough** search of the neighborhood.
- a **more precise** estimation of the neighborhood quality.



X/X Ratio in metaheuristics

Assuming a constant number *n* of samples evaluated in each neighborhood, a **smaller** search radius L_{max} results in...

- a **smaller range**, i.e., a smaller number of points which are potentially considered for evaluation.
- a **higher risk** of not finding a promising point for the next global jump and getting stuck in a local optimum.



X/X Ratio in metaheuristics

With a **smaller** radius...

- the samples are picked mainly from the **nearby area** of the current position and...
- are **very similar** to the current point (SSA!).
- Therefore a smaller radius is particularly reasonable if the current solution is already **very good**, i.e. very close to the global optimum.

X/X Ratio in metaheuristics

With a **larger** radius...

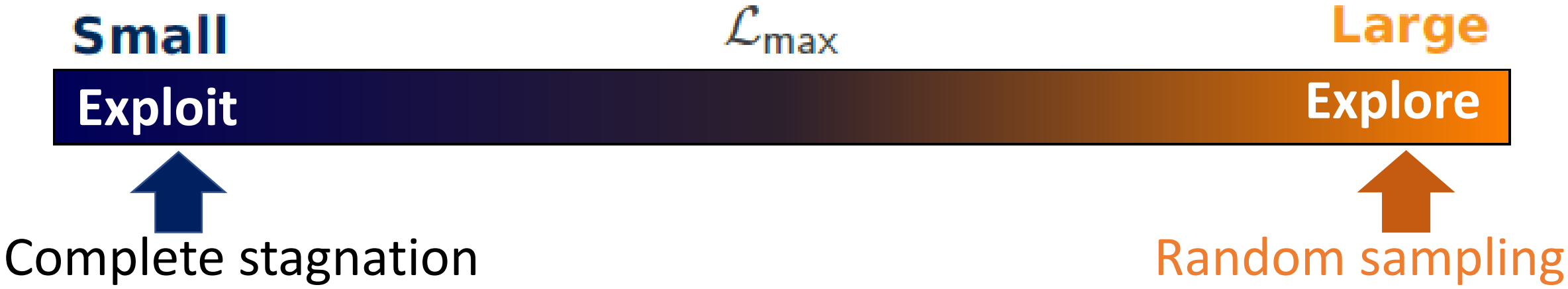
- the samples are picked mainly from the area **further away** from the current position and...
- are **substantially different** from the current point (SSA!).

Therefore, a larger radius is particularly reasonable if there is a **lack of evidence** to suggest that the current solution is **particularly good**.

X/X Ratio in metaheuristics

- The appropriate search radius is dictated by the **available information** about the quality of the known solutions relatively to the global optimum.
- In situations with a **small** amount of available information, the confidence in the quality of the known solutions is **low** , so that the **exploration** of yet not considered solutions (within a **larger** radius around the current position) is prioritized.

X/X Ratio in metaheuristics



The concepts of **exploration** and **exploitation** of information are highly relevant not only for the random search, but for most optimization metaheuristics and even for approaches from other domains, e.g., reinforcement learning.

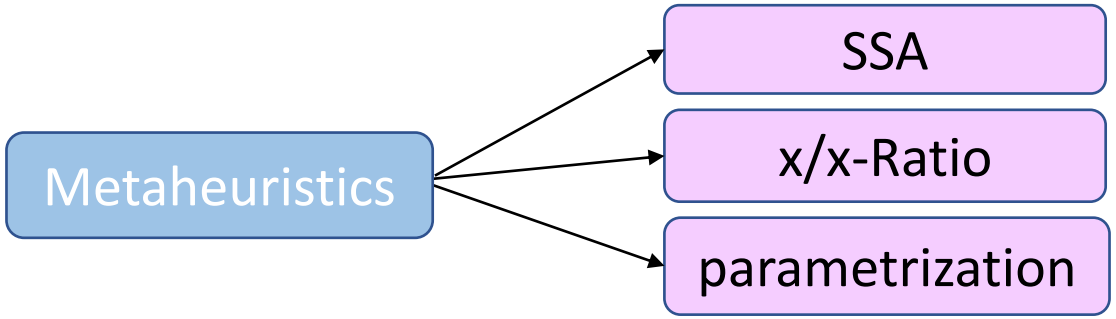
Finding the right **ratio** between **exploration** and **exploitation** is one of the biggest challenges during the design of optimization metaheuristics.

X/X Ratio in metaheuristics

- **Q.: How to determine the 'right' search radius for a given problem?**
- Unfortunately, there is no approach able to analytically determine the optimal X/X Ratio (while staying largely problem-independent).
- However, interpreting the optimization as a gradual accumulation of problem information enables a formulation of general X/X strategies.
- In many metaheuristics, the X/X-ratio...
- is often changed throughout the course of the optimization.
  - is set such that the optimizer focuses on exploration in the early and on
  - exploitation in the late stages of the optimization (*optimism in the face of uncertainty*).

X/X Ratio in metaheuristics

- Considering the random search, there are several other parameters beside the search radius which have a strong influence on the course of the optimization:
  - The number of samples evaluated in each neighborhood.
  - The assumptions about the quality distribution (used for the sampling procedure).
- Similarly to the radius, there is no analytical approach capable of finding the ideal set of parameters.
- Since the right parameter set depends on the characteristics of the search space, the parametrization of a metaheuristic corresponds to a *guess* about what the search space of the unknown problem will look like.



Selecting the optimizer

Selecting the optimizer-general considerations

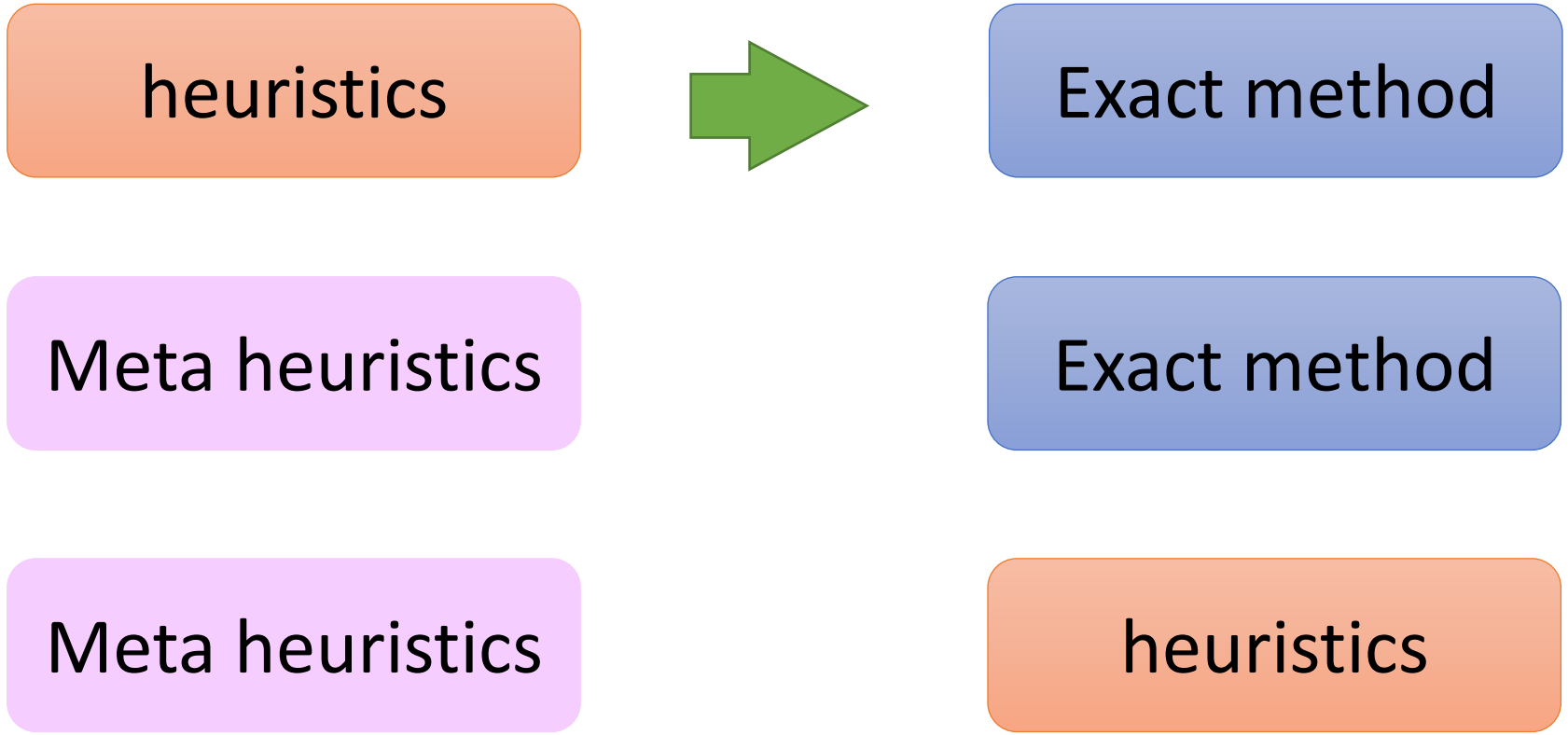
When choosing the right optimization approach, the three main considerations are:

- The *'average'* expected result **quality**.
- The (*'average'* expected) optimization **time**.
- The required **insight** into...
  - the problem.
  - the optimization approach.

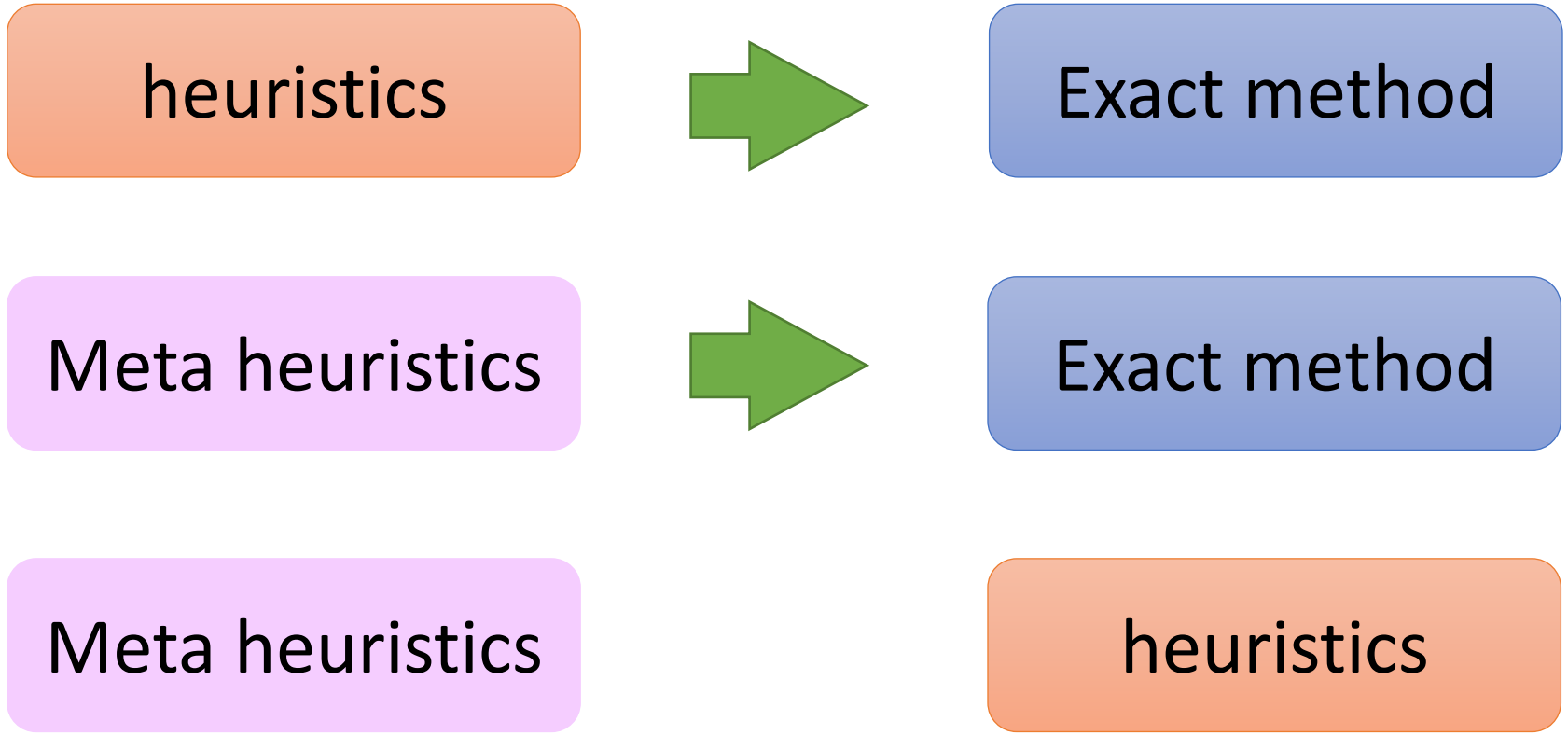
Selecting the optimizer-Result quality

heuristics	Exact method
Meta heuristics	Exact method
Meta heuristics	heuristics

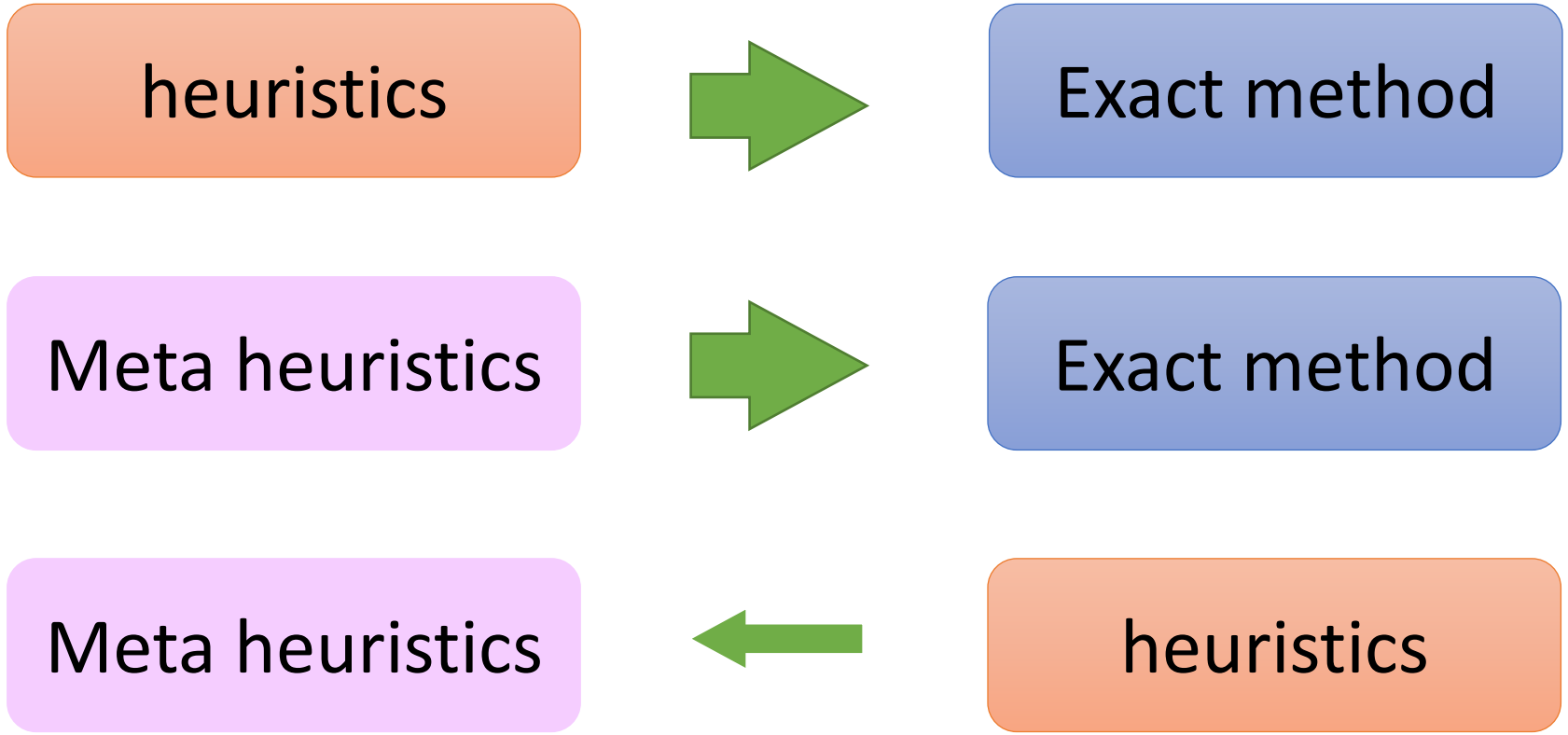
Selecting the optimizer-Result quality



Selecting the optimizer-Result quality



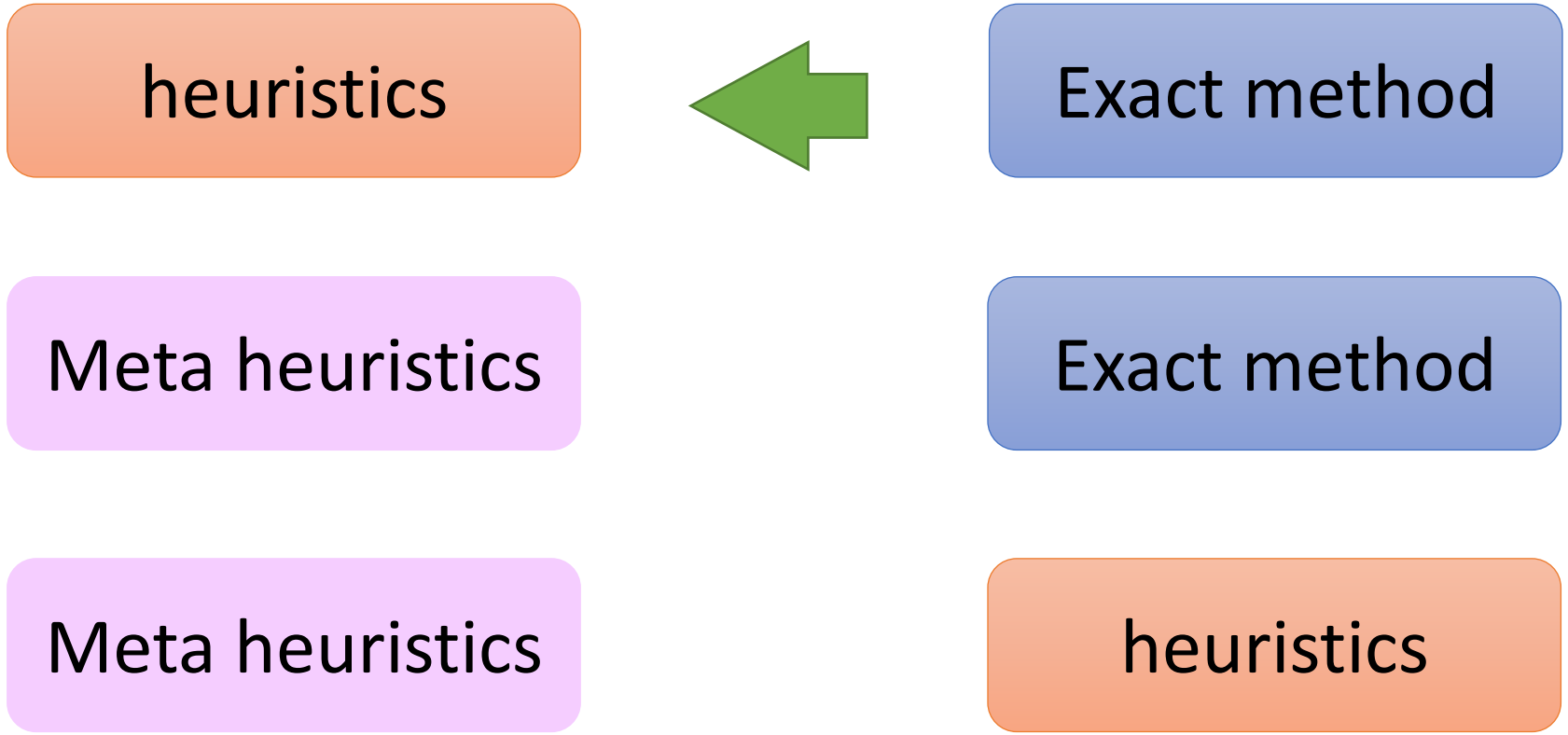
Selecting the optimizer-Result quality



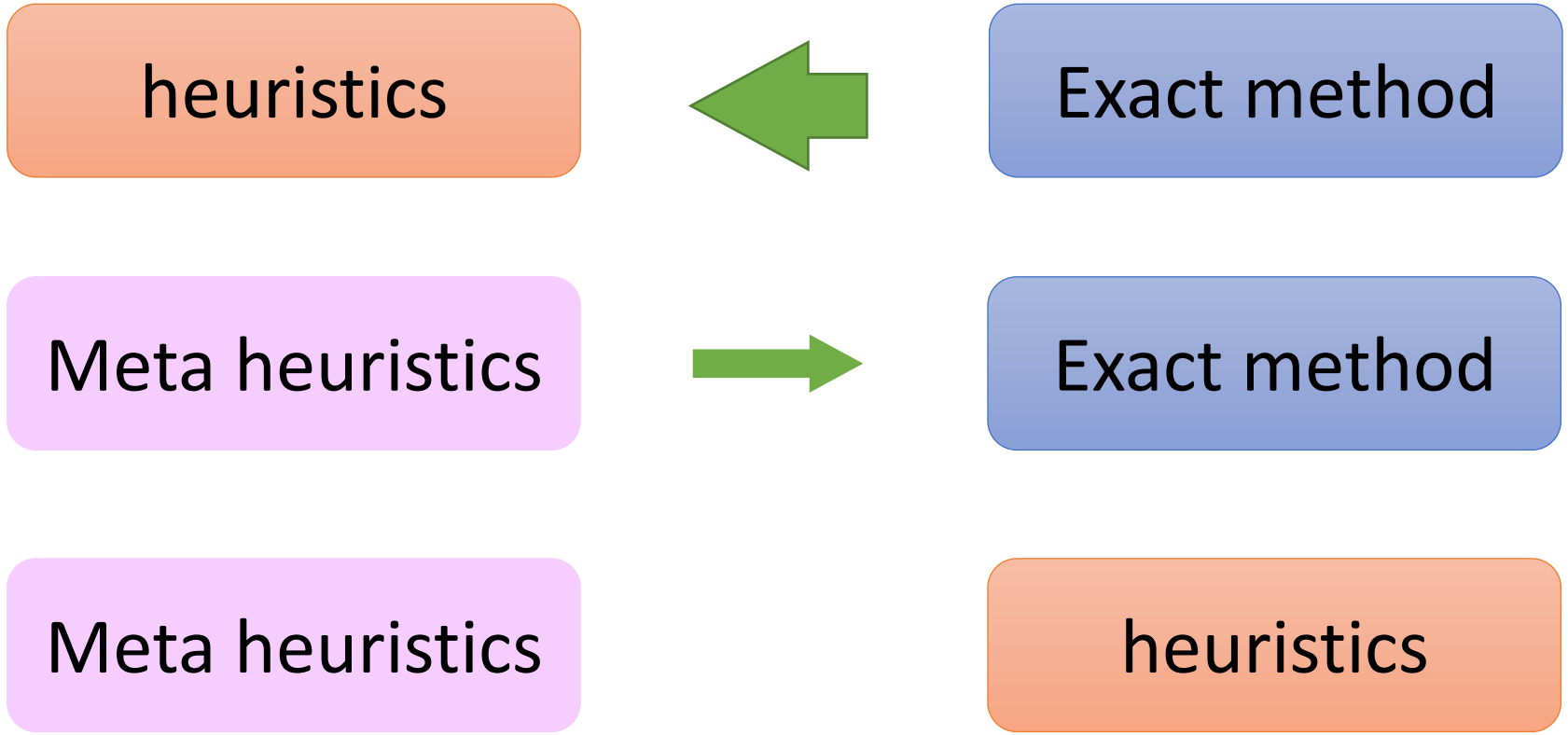
Selecting the optimizer- optimization time

heuristics	Exact method
Meta heuristics	Exact method
Meta heuristics	heuristics

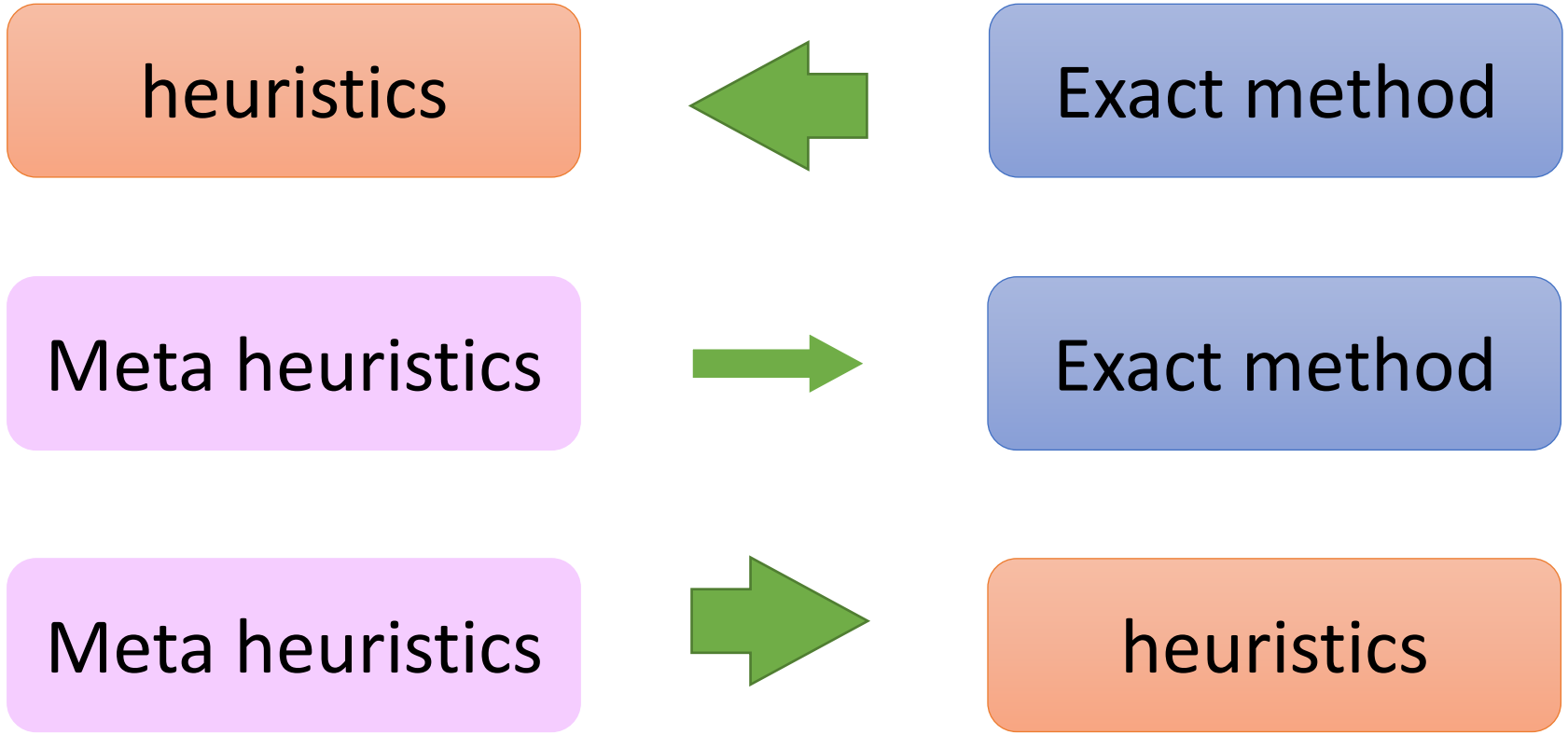
Selecting the optimizer- optimization time



Selecting the optimizer- optimization time



Selecting the optimizer- optimization time



Metaheuristics vs. closed form

Metaheuristics **VS.** Exact method

- As a rule of thumb, you should always resort to closed form approaches **if they are applicable to your problem**.
- From the perspective of the system designer, these two types of optimizers offer different trade-offs:

	Closed form	**Metaheuristics**
Problem formulation	**Hard**	**Easy**
Solving the problem	**Easy**	**Hard**

Metaheuristics vs. heuristics

Metaheuristics **VS.** heuristics

- Compared to heuristics, metaheuristics have the advantages of...
  - not restricting the search space.
  - providing solutions which are at least as good.
  - being very flexible considering changing problems.
- For an optimization problem, heuristics are chosen if...
  - the time (and the computation power) available for the optimization is limited.
  - the search space is exceedingly large and cannot be searched without applying severe restrictions.
  - problem-specific knowledge is available.
  - the key features of the problem are likely to remain unchanged for a long time.

Metaheuristics vs. heuristics

Metaheuristics **VS.** heuristics

Message routing in an automotive network.

- Many different objectives.
- Untrivial design decision implications.
- Continuously changing problem conditions due to
  - technological innovations
  - new business models

Movement of a laboratory placement robot.

- Required time as the only objective.
- Clear consequences of design decisions.
- "Stable" problem.

universität innsbruck

Thank you for your attention☺

Dr. Zahra Najafabadi Samani

Email: Zahra.Najafabadi-Samani@uibk.ac.at