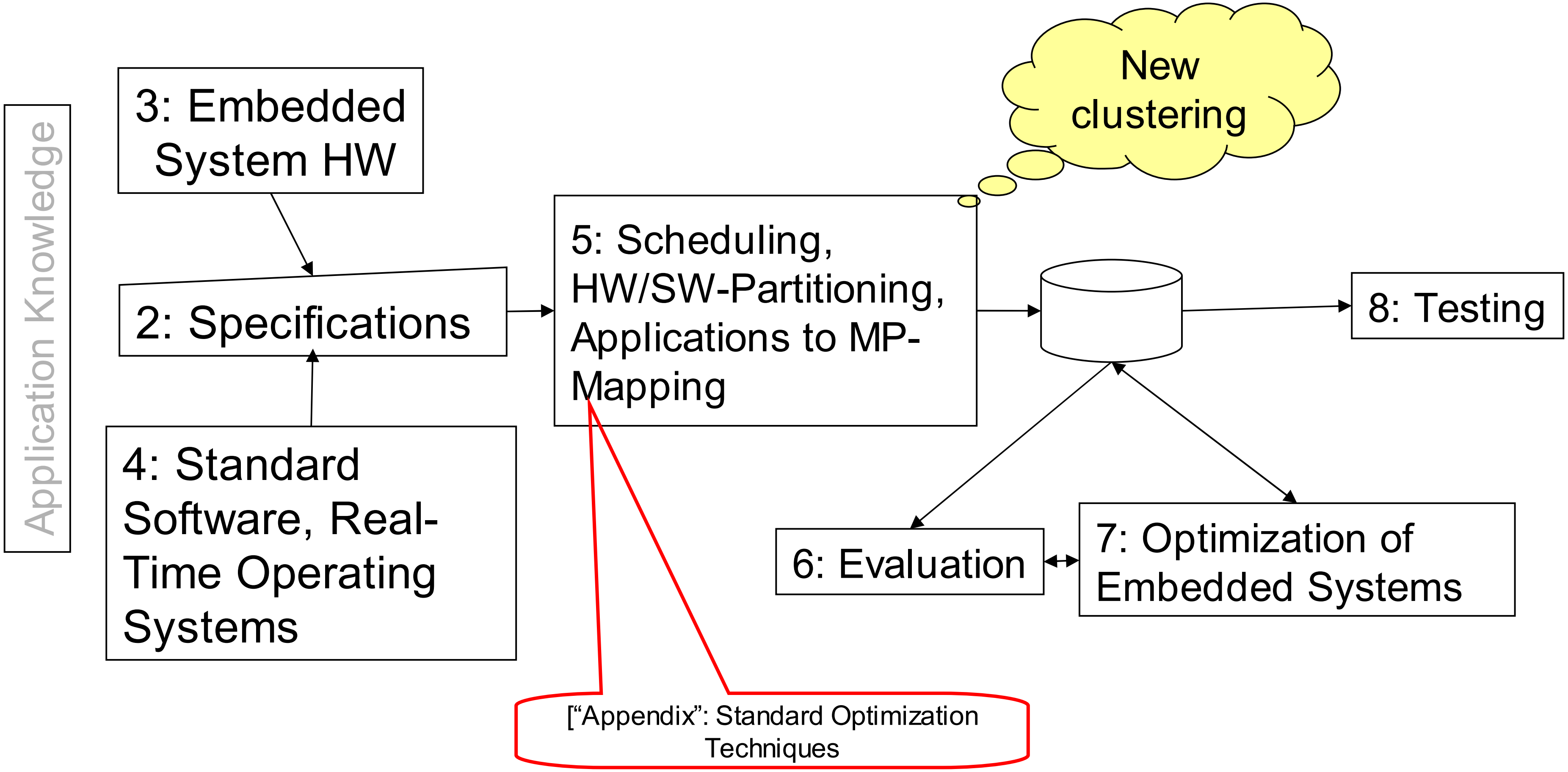
Standard Optimization Techniques

Peter Marwedel
Informatik 12
TU Dortmund
Germany

2008/12/06

Graphics: © Alexandra Nolte, Gesine Marwedel, 2003

Structure of this course



Optimization Alternatives

- Use of *classical single objective optimization* methods
  - simulated annealing, tabu search
  - integer linear program
  - other constructive or iterative heuristic methods
- *Decision making* (weighting the different objectives) is done *before the optimization*.

- *Population based optimization methods*
  - evolutionary algorithms
  - genetic algorithms
- *Decision making* is done *after the optimization*.

Integer programming models

Ingredients:

- Cost function
- Constraints

$\Big\}$ Involving linear expressions of integer variables from a set *X*

Cost function $$C = \sum_{x_i \in X} a_i x_i \text{ with } a_i \in R, x_i \in \mathbb{N} \quad (1)$$

Constraints: $$\forall j \in J : \sum_{x_i \in X} b_{i,j}\, x_i \geq c_j \text{ with } b_{i,j}, c_j \in \mathbb{R} \quad (2)$$

Def.: The problem of minimizing (1) subject to the constraints (2) is called an **integer (linear) programming (ILP) problem**.

If all x_i are constrained to be either 0 or 1, the IP problem said to be a **0/1 integer (linear) programming problem**.

Example

$$C = 5x_1 + 6x_2 + 4x_3$$

$$x_1 + x_2 + x_3 \geq 2$$

$$x_1, x_2, x_3 \in \{0,1\}$$

x_1	x_2	x_3	C
0	1	1	10
1	0	1	9
1	1	0	11
1	1	1	15

← Optimal

Remarks on integer programming

- Maximizing the cost function: just set C'=-C
- Integer programming is NP-complete.
- Running times depend exponentially on problem size, but problems of >1000 vars solvable with good solver (depending on the size and structure of the problem)
- The case of $x_i \in \mathbb{R}$ is called *linear programming* (LP). LP has polynomial complexity, but most algorithms are exponential, still in practice faster than for ILP problems.
- The case of some $x_i \in \mathbb{R}$ and some $x_i \in \mathbb{N}$ is called *mixed integer-linear programming.*
- ILP/LP models can be a good starting point for modeling, even if in the end heuristics have to be used to solve them.

Simulated Annealing

- General method for solving combinatorial optimization problems.
- Based the model of slowly cooling crystal liquids.
- Some configuration is subject to changes.
- Special property of Simulated annealing: Changes leading to a poorer configuration (with respect to some cost function) are accepted with a certain probability.
- This probability is controlled by a temperature parameter: the probability is smaller for smaller temperatures.

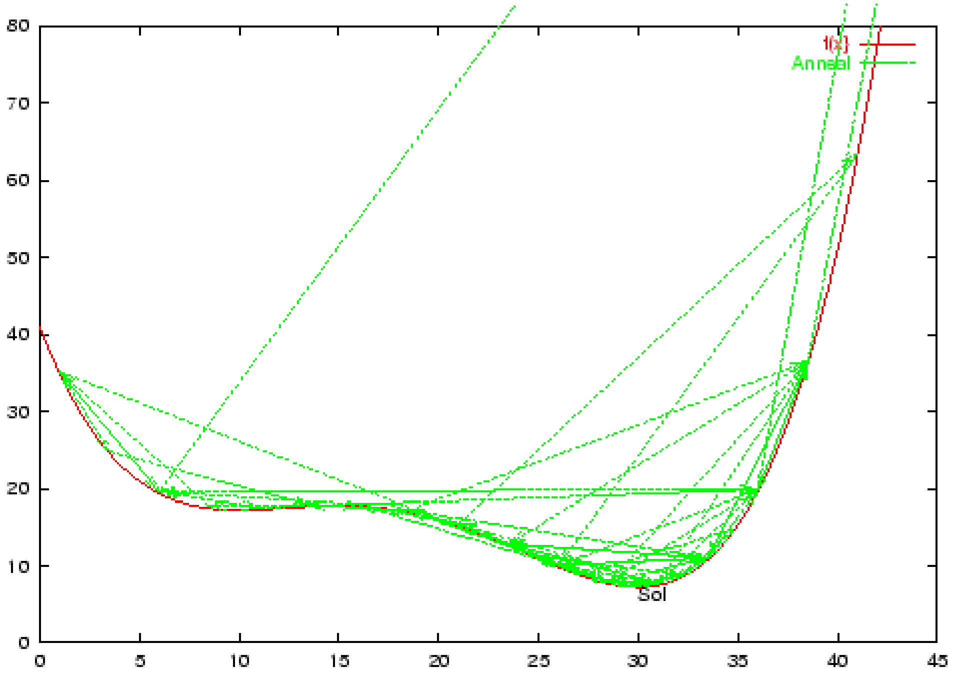
Simulated Annealing Algorithm

```
procedure SimulatedAnnealing;
var i, T: integer;
 begin
   i := 0; T := MaxT;
   configuration:= <some initial configuration>;
   while not terminate(i, T) do
    begin
     while InnerLoop do
       begin NewConfig := variation(configuration);
         delta := evaluation(NewConfig,configuration);
         if delta < 0
         then configuration := NewConfig;
         else if SmallEnough(delta, T, random(0,1))
           then configuration := Newconfiguration;
       end;
   T:= NewT(i,T); i:=i+1;
 end; end;
```

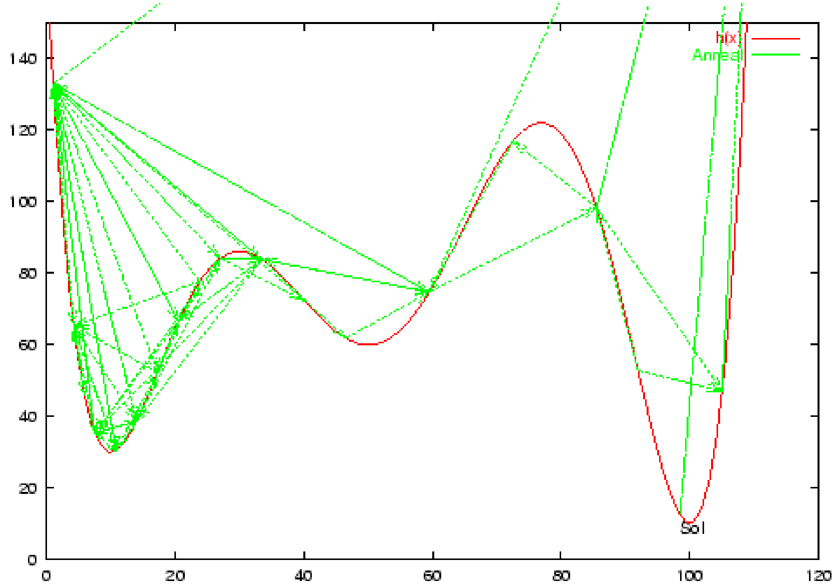
Explanation

- Initially, some random initial configuration is created.
- Current temperature is set to a large value.
- Outer loop:
  - Temperature is reduced for each iteration
  - Terminated if (temperature ≤ lower limit) or (number of iterations ≥ upper limit).
- Inner loop: For each iteration:
  - New configuration generated from current configuration
  - Accepted if (new cost ≤ cost of current configuration)
  - Accepted with temperature-dependent probability if (cost of new config. > cost of current configuration).

Behavior for actual functions



130 steps



200 steps

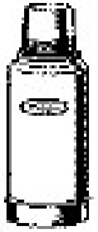
[people.equars.com/~marco/poli/phd/node57.html]

http://foghorn.cadlab.lafayette.edu/cadapplets/fp/fpIntro.html

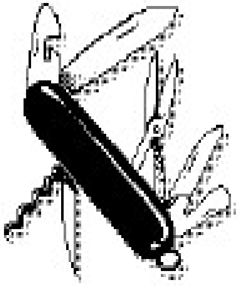
The Knapsack Problem

weight = 750g
profit = 5

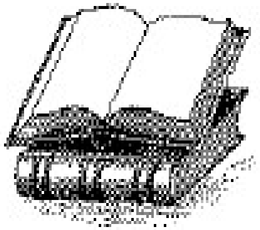
weight = 1500g
profit = 8



weight = 300g
profit = 7



weight = 1000g
profit = 3

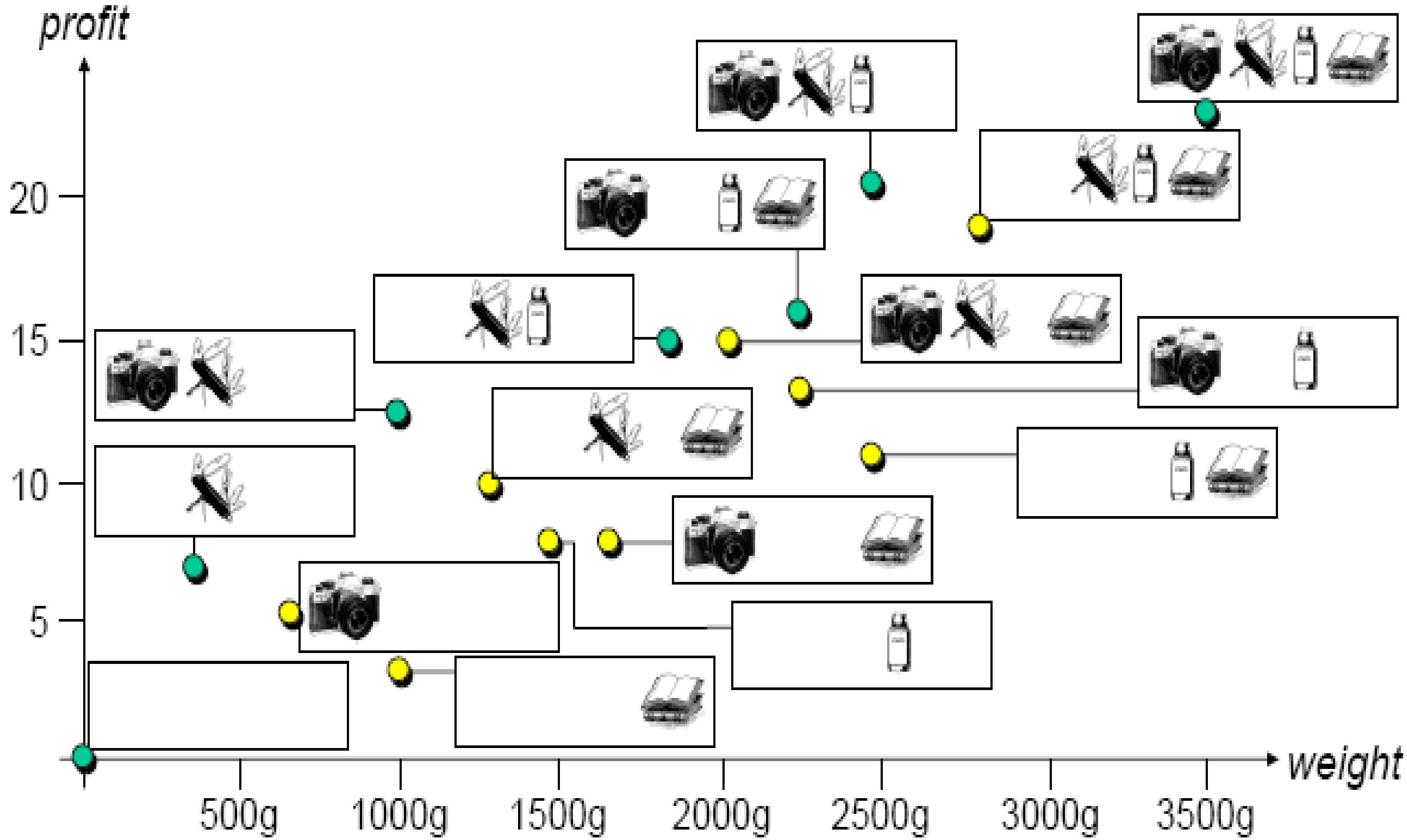


?

Goal: choose subset that

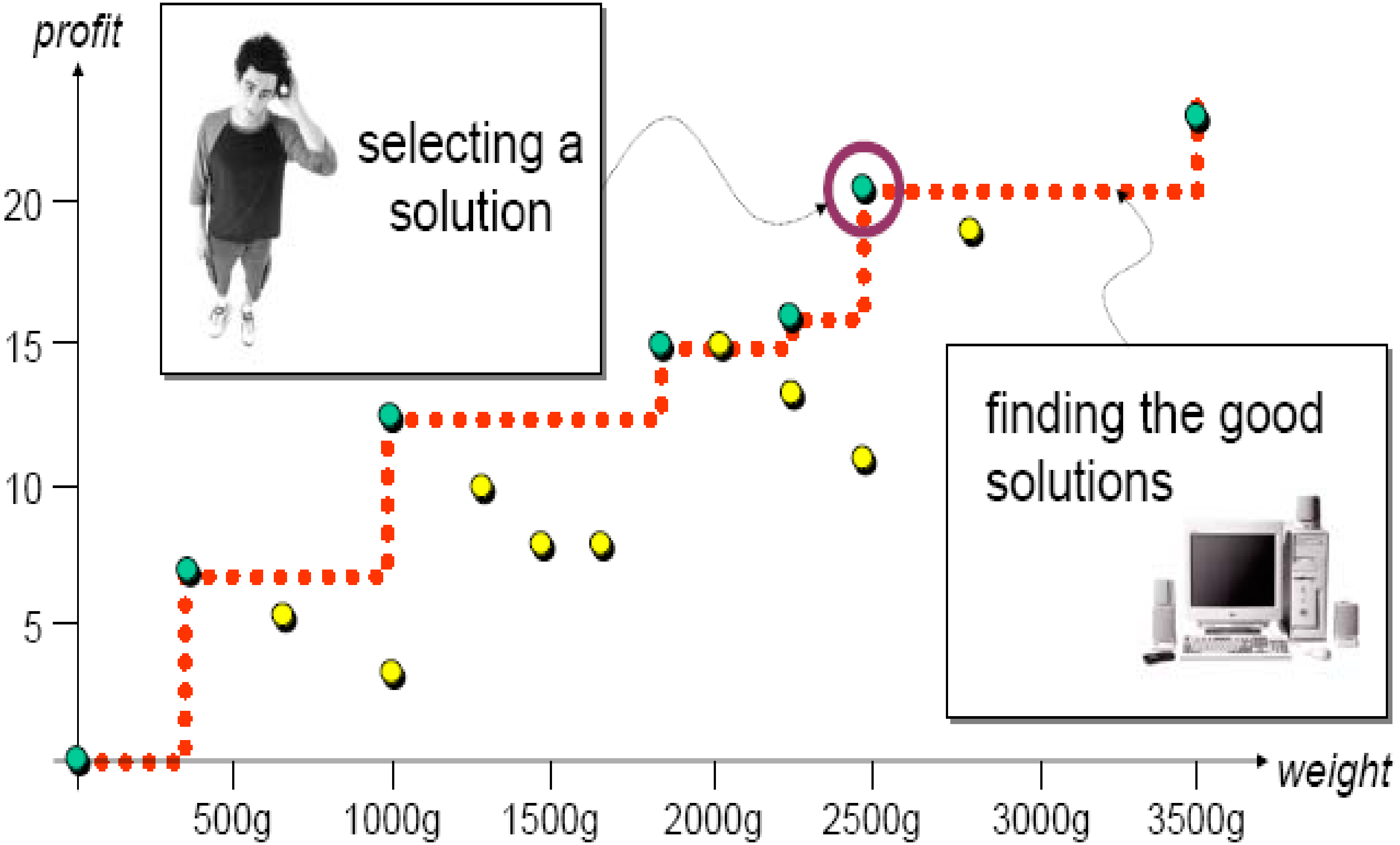
- maximizes overall profit
- minimizes total weight

The Solution Space



The Trade-off Front

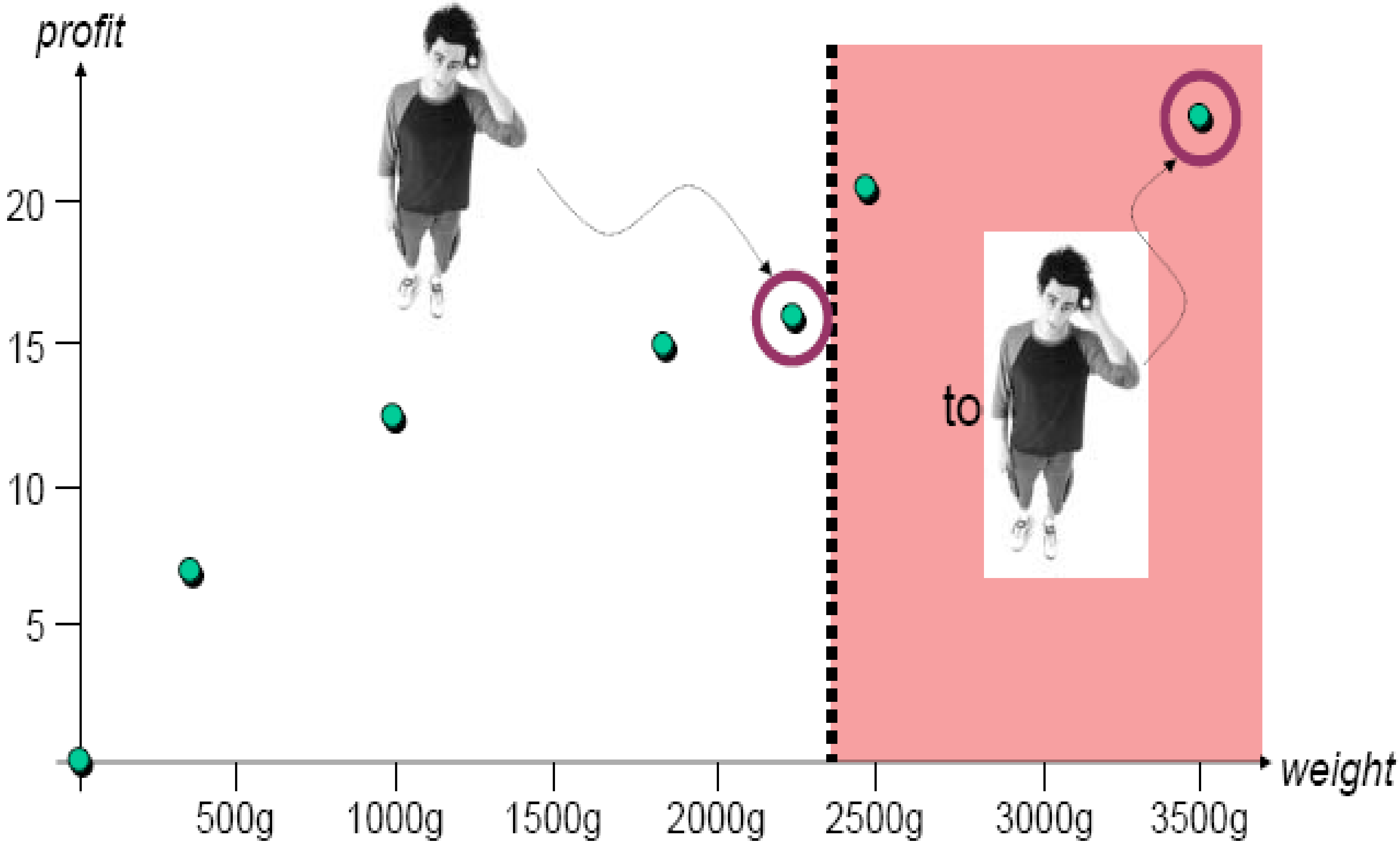
Observations: ❶ there is no single optimal solution, but
❷ some solutions (🟢) are better than others (🟡)



Decision Making: Selecting a Solution

Approaches:

- profit more important than cost (ranking)
- weight must not exceed 2400g (constraint)



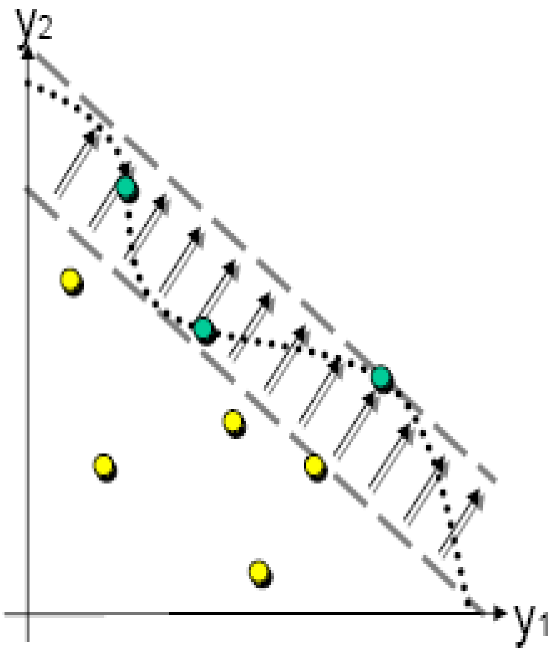
Optimization Alternatives

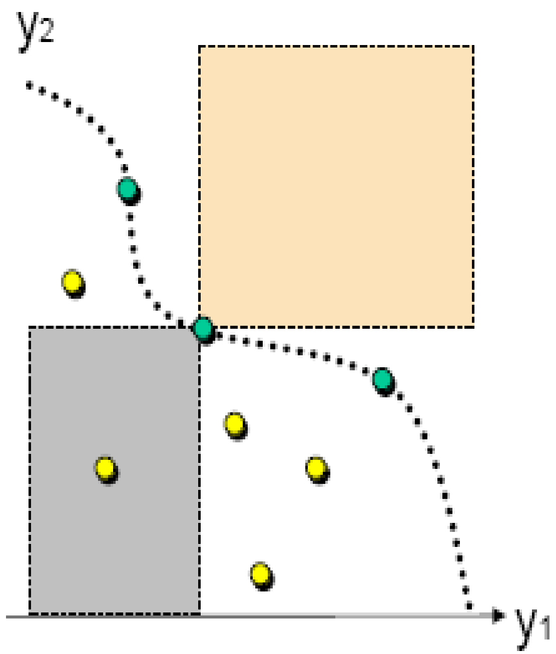
scalarization

weighted sum

population-based

SPEA2

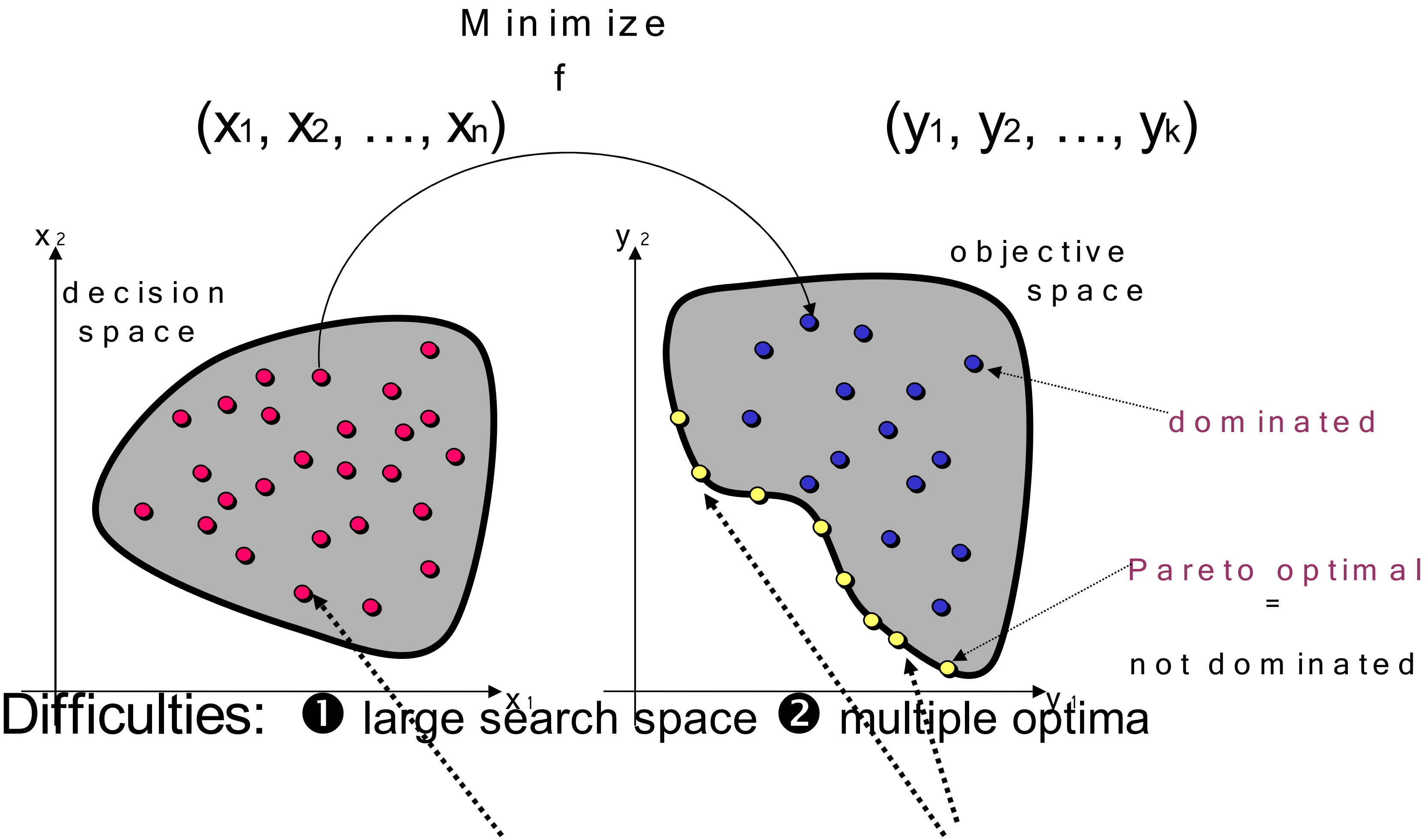




parameter-oriented
scaling-dependent

set-oriented
scaling-independent

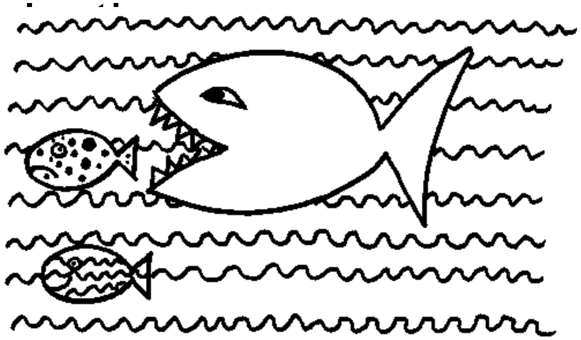
Multiobjective Optimization



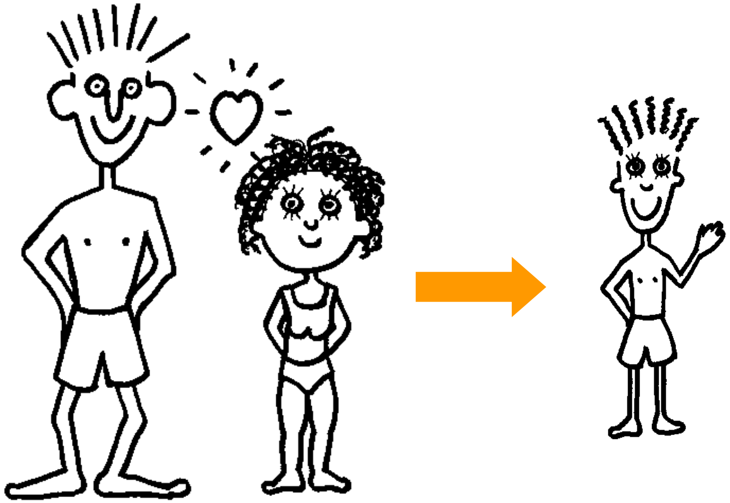
Difficulties: ❶ large search space ❷ multiple optima

Principles of Evolution

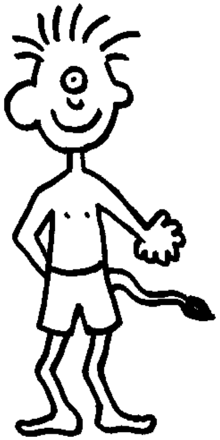
❶ S



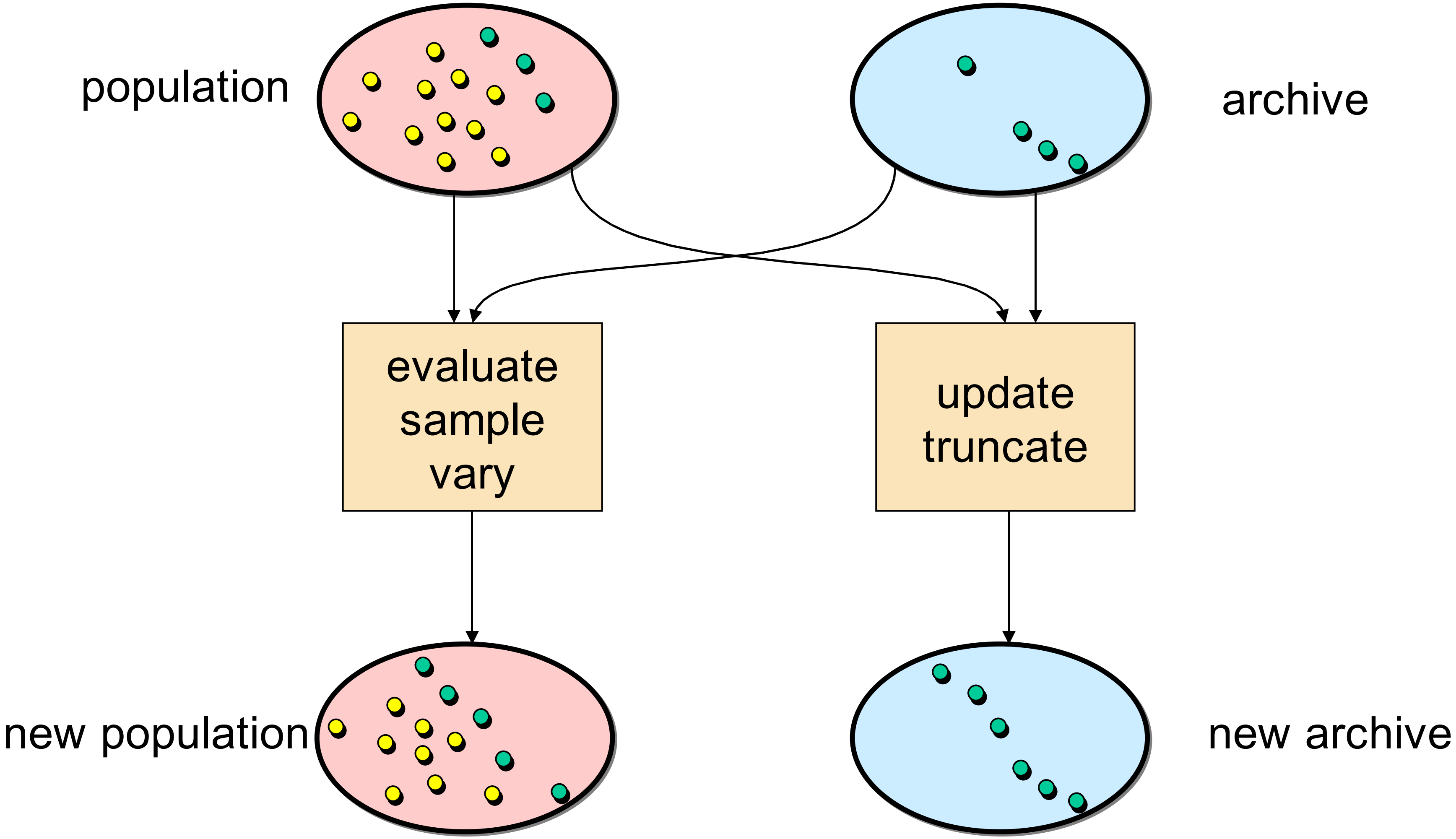
❷ Cross-over



❸ Mutation



A Generic Multiobjective EA



population

archive

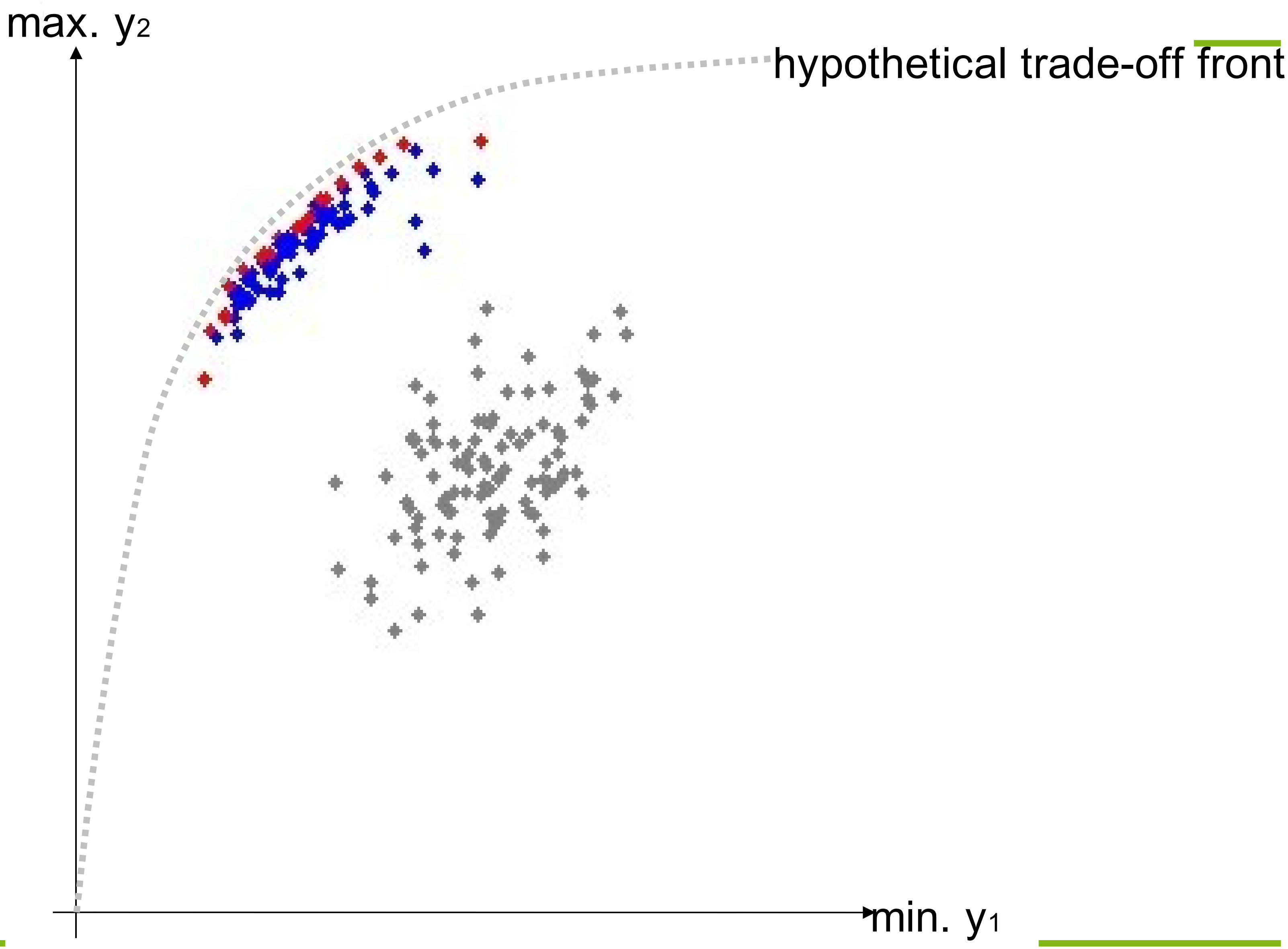
evaluate
sample
vary

update
truncate

new population

new archive

An Evolutionary Algorithm in Action



Dominance, Pareto Points

- A (design) point J_k is *dominated* by J_i, if J_i is
  - better or equal than J_k in all criteria and
  - better in at least one criterion.

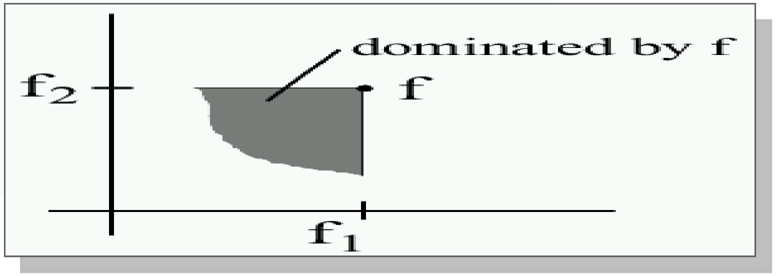
- A point is Pareto-optimal or a *Pareto-point,* if it is not dominated.

- The domination relation imposes a partial order on all design points
  - We are faced with a set of optimal solutions.
  - Divergence of solutions vs. convergence.

Multi-objective Optimization

Definition 1 (Dominance relation)
Let $f, g \in \mathbb{R}^m$*. Then* f *is said to dominate* g*, denoted as* $f \succ g$*, iff*

1. $\forall i \in \{1, \ldots, m\} : f_i \geq g_i$
2. $\exists j \in \{1, \ldots, m\} : f_j > g_j$

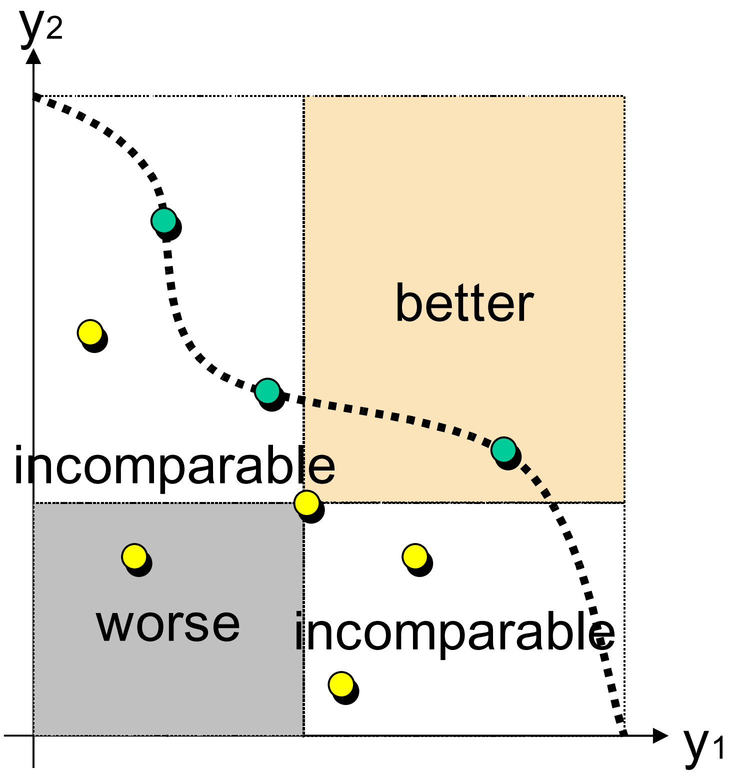


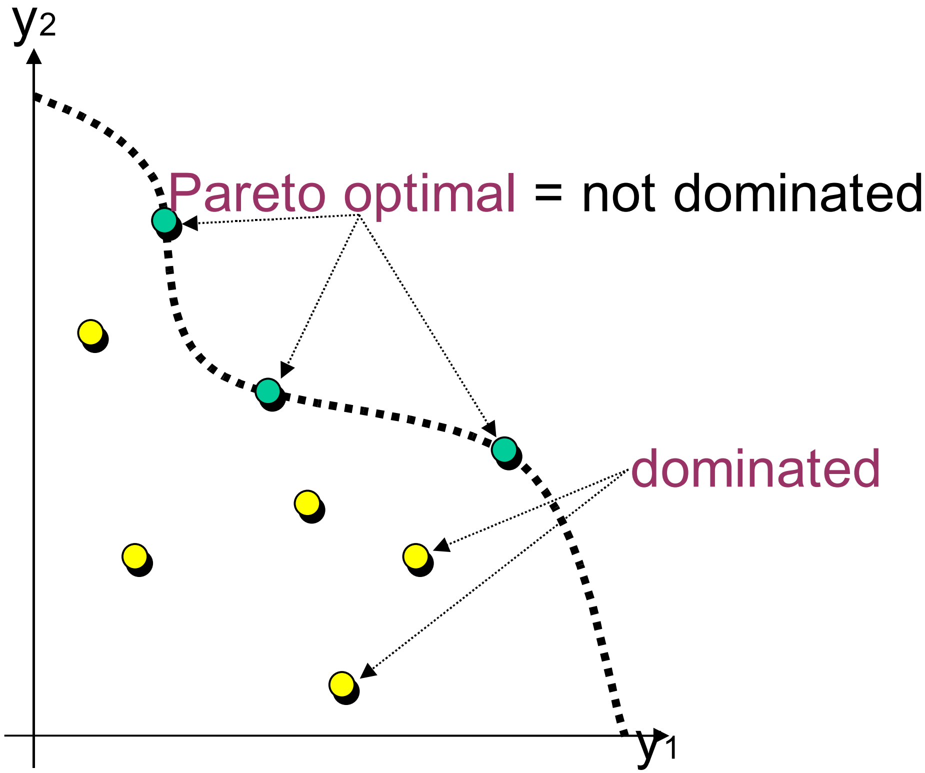
Definition 2 (Pareto set)
Let $F \subseteq \mathbb{R}^m$ *be a set of vectors. Then the Pareto set* $F^* \subseteq F$ *is defined as follows:* F^* *contains all vectors* $g \in F$ *which are not dominated by any vector* $f \in F$*, i.e.*

$$F^* := \{g \in F \mid \nexists f \in F : f \succ g\} \tag{1}$$

Multiobjective Optimization

Maximize $(y_1, y_2, \ldots, y_k) = f(x_1, x_2, \ldots, x_n)$





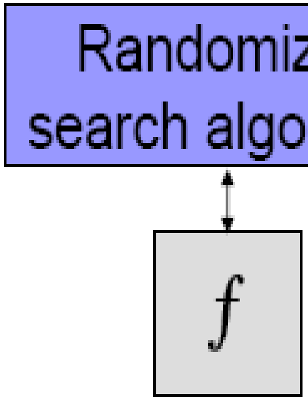
Pareto set = set of all Pareto-optimal solutions

Randomized (Black Box) Search Algorithms

Idea: find good solutions without investigating all solutions

Assumptions: better solutions can be found in the neighborhood of good solutions

information available only by function evaluations



Randomized search algorithm

f

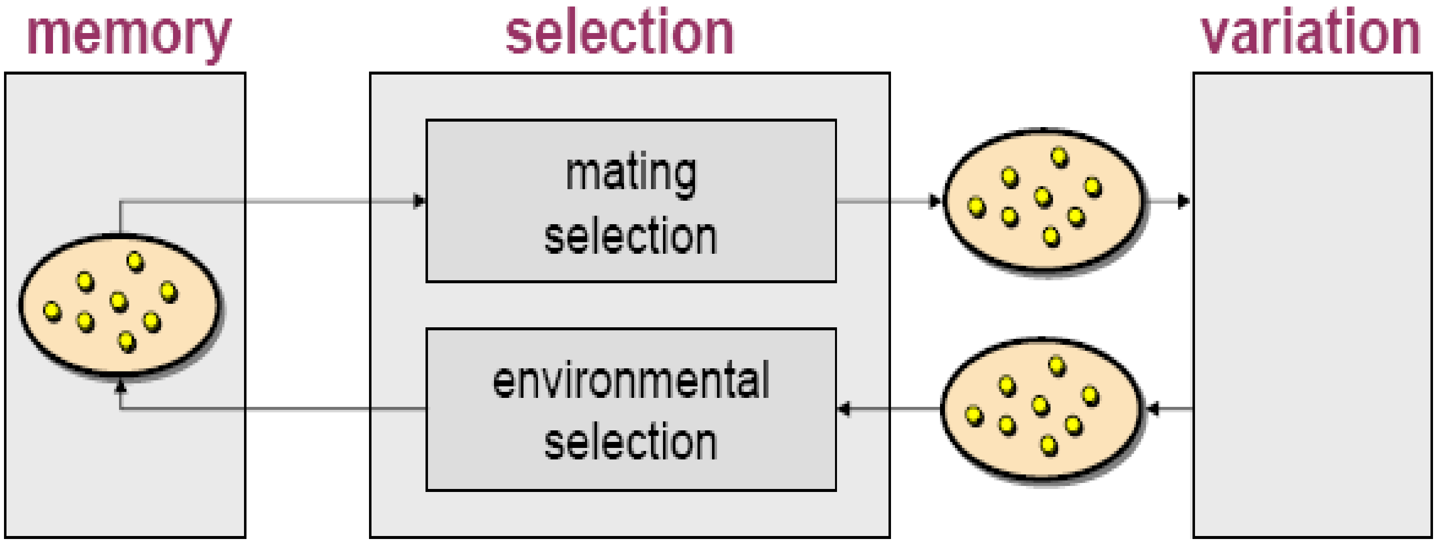
$t = 1$:

(randomly) choose a solution x_1 to start with

$t \geq t+1$:

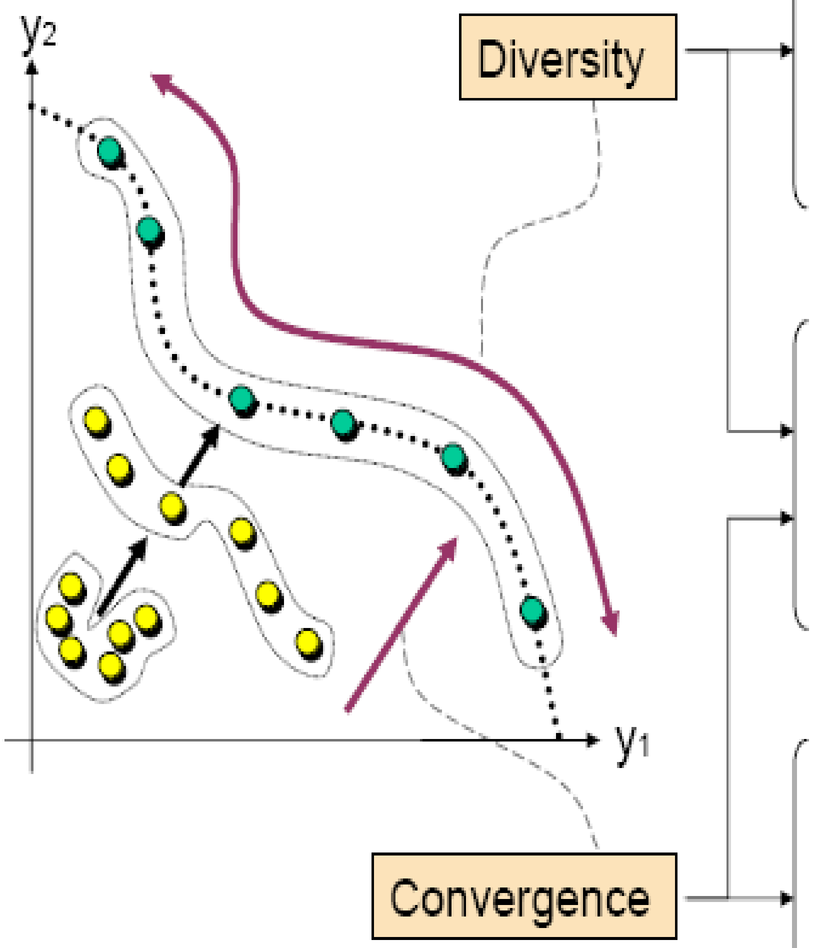
(randomly) choose a solution x_{t+1} using solutions $x_1, \ldots, x_t$

Types of Randomized Search Algorithms



EA evolutionary algorithm	≥ 1	both
TS tabu search	1	no mating selection
SA simulated annealing	1	no mating selection

Issues in Multi-Objective Optimization



- How to maintain a diverse Pareto set approximation?

  ❷ density estimation

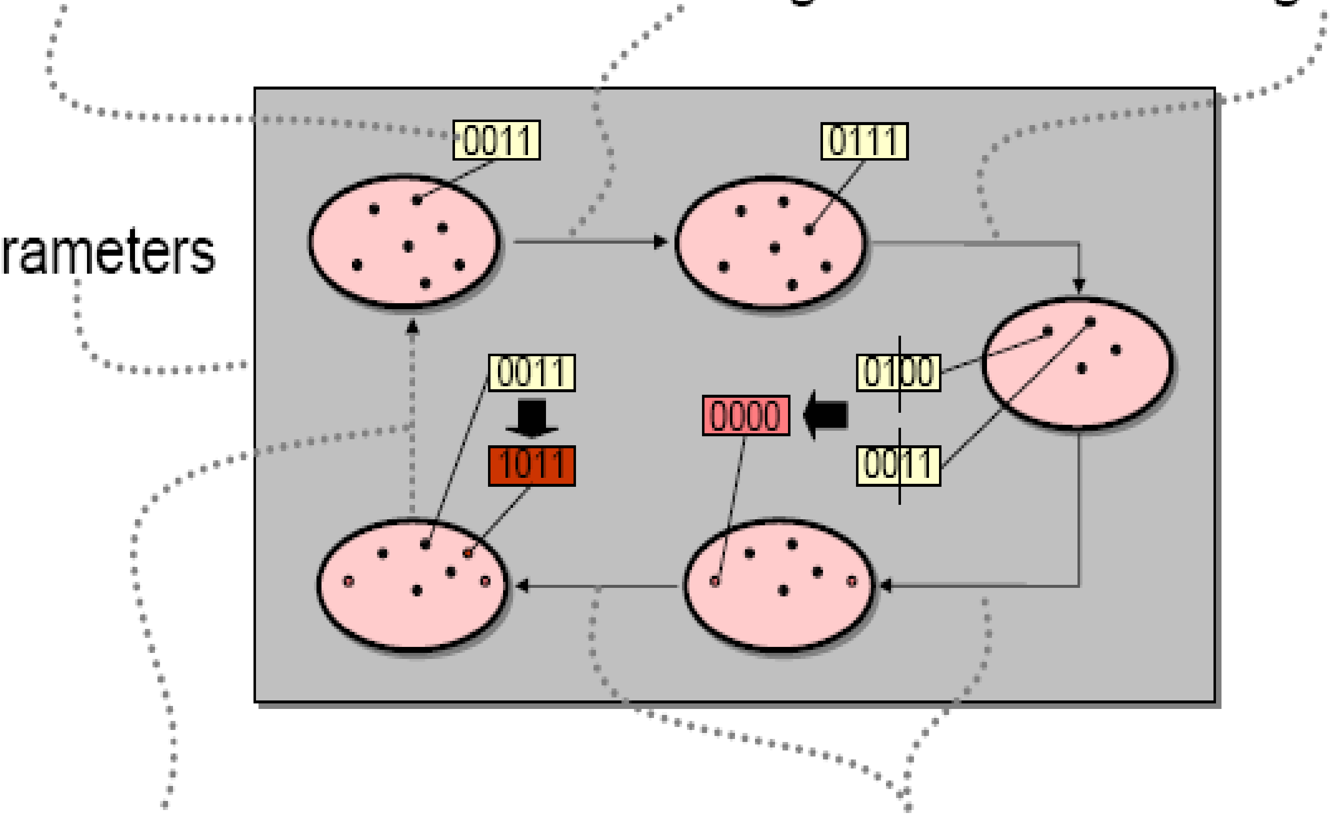
- How to prevent nondominated solutions from being lost?

  ❸ environmental selection

- How to guide the population towards the Pareto set?

  ❶ fitness assignment

Design Choices



representation

fitness assignment

mating selection

parameters

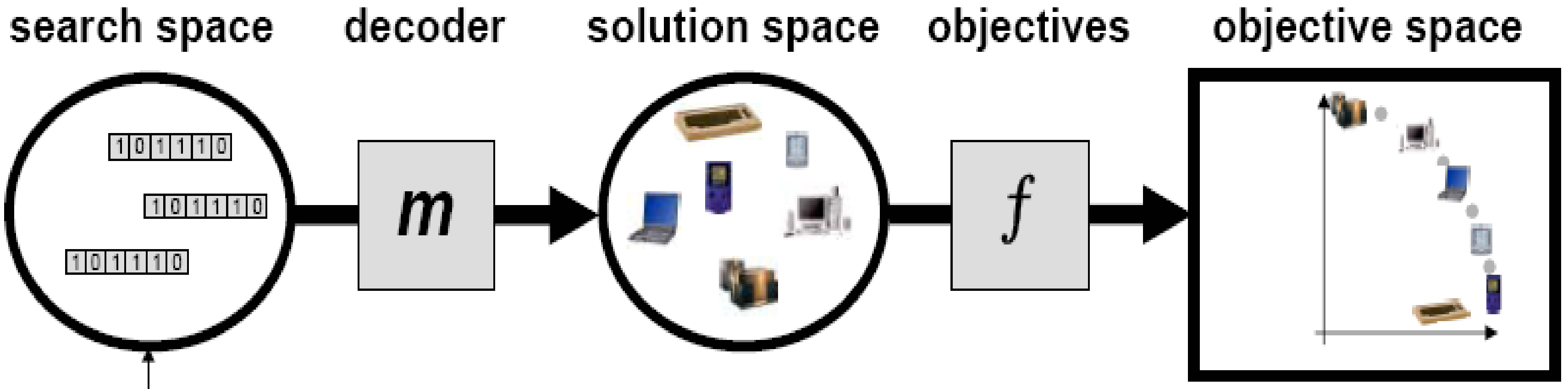
environmental selection

variation operators

Example: SPEA2 Algorithm

Step 1:	Generate initial population P0 and empty archive (external set) A_0. Set $t = 0$.
Step 2:	Calculate fitness values of individuals in P_t and A_t.
Step 3:	A_{t+1} = nondominated individuals in P_t and A_t. If size of $A_{t+1} > N$ then reduce A_{t+1}, else if size of $A_{t+1} < N$ then fill A_{t+1} with dominated individuals in P_t and A_t.
Step 4:	If $t > T$ then output the nondominated set of A_{t+1}. Stop.
Step 5:	Fill mating pool by binary tournament selection.
Step 6:	Apply recombination and mutation operators to the mating pool and set P_{t+1} to the resulting population. Set $t = t + 1$ and go to Step 2.

Representation



search space decoder solution space objectives objective space

solutions encoded by $\underbrace{\text{vectors, matrices}}_{\text{fixed length}}$, $\underbrace{\text{trees, lists}}_{\text{variable length}}$, ...

Issues:

- completeness (each solution has an encoding)
- uniformity (all solutions are represented equally)
- redundancy (cardinality of search space vs. solution space)
- feasibility (each encoding maps to a feasible solution)

Summary

Single objective optimization methods

- decision is performed during optimization
- Examples: integer programming, simulated annealing

Multiple objective optimization methods

- decision is done after optimization
- Example: Evolutionary algorithms
- Refer to publications of Thiele or Schwefel et al. for more information

Concept of Pareto points

- eliminates large set of non-relevant design points
- allows separating optimization and decision