4. Design Space Exploration of Embedded Systems

© Lothar Thiele

ETH Zurich, Switzerland

Contents of Lectures (Lothar Thiele)

1. Introduction to Embedded System Design
2. Software for Embedded Systems
3. Real-Time Scheduling
4. Design Space Exploration
5. Performance Analysis

Topics

- ***Multi-Objective Optimization***
  - ***Introduction***
  - Multi-Objective Optimization
  - Algorithm Design
    - Design Choices
    - Representation
    - Fitness and Selection
    - Variation Operators
  - Implementation
- Design Space Exploration
  - System Design
  - Problem Specification
  - Example

Evolutionary Multiobjective Optimization Algorithms

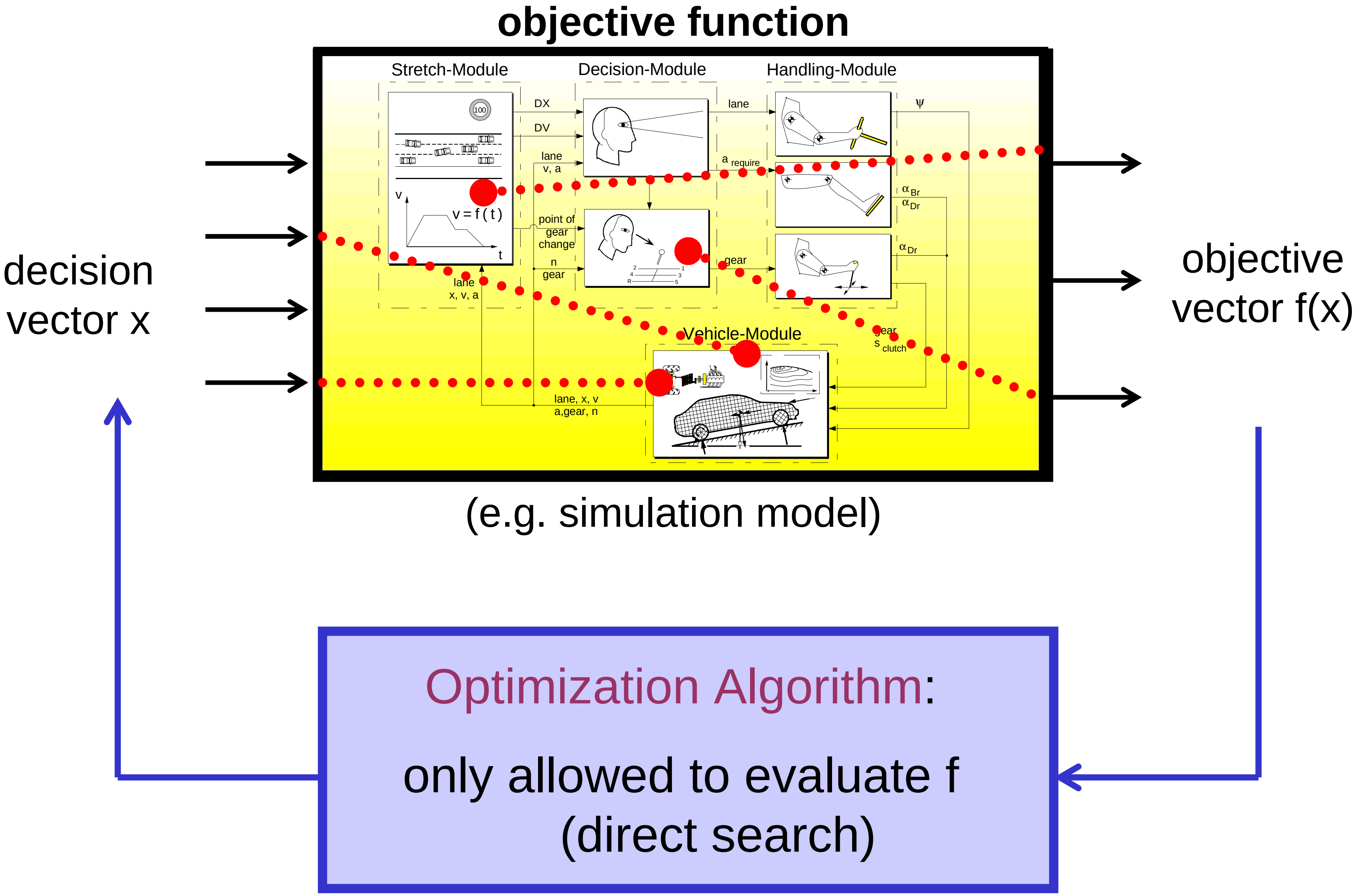
What are Evolutionary Algorithms?

- randomized, **problem-independent** search heuristics
  → applicable to black-box optimization problems

How do they work?

- by iteratively improving a **population** of solutions by variation and selection
  → can find many different optimal solution in a single run

Black-Box Optimization



objective function

decision vector x

objective vector f(x)

(e.g. simulation model)

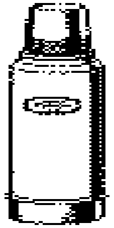
Optimization Algorithm:

only allowed to evaluate f
(direct search)

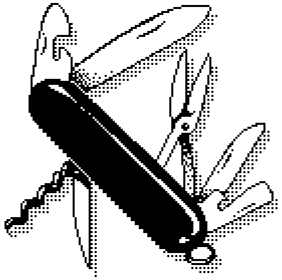
The Knapsack Problem

weight = 750g
profit = 5

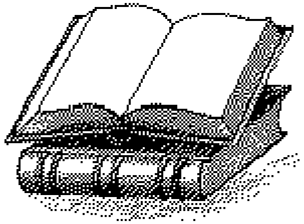
weight = 1500g
profit = 8



weight = 300g
profit = 7



weight = 1000g
profit = 3

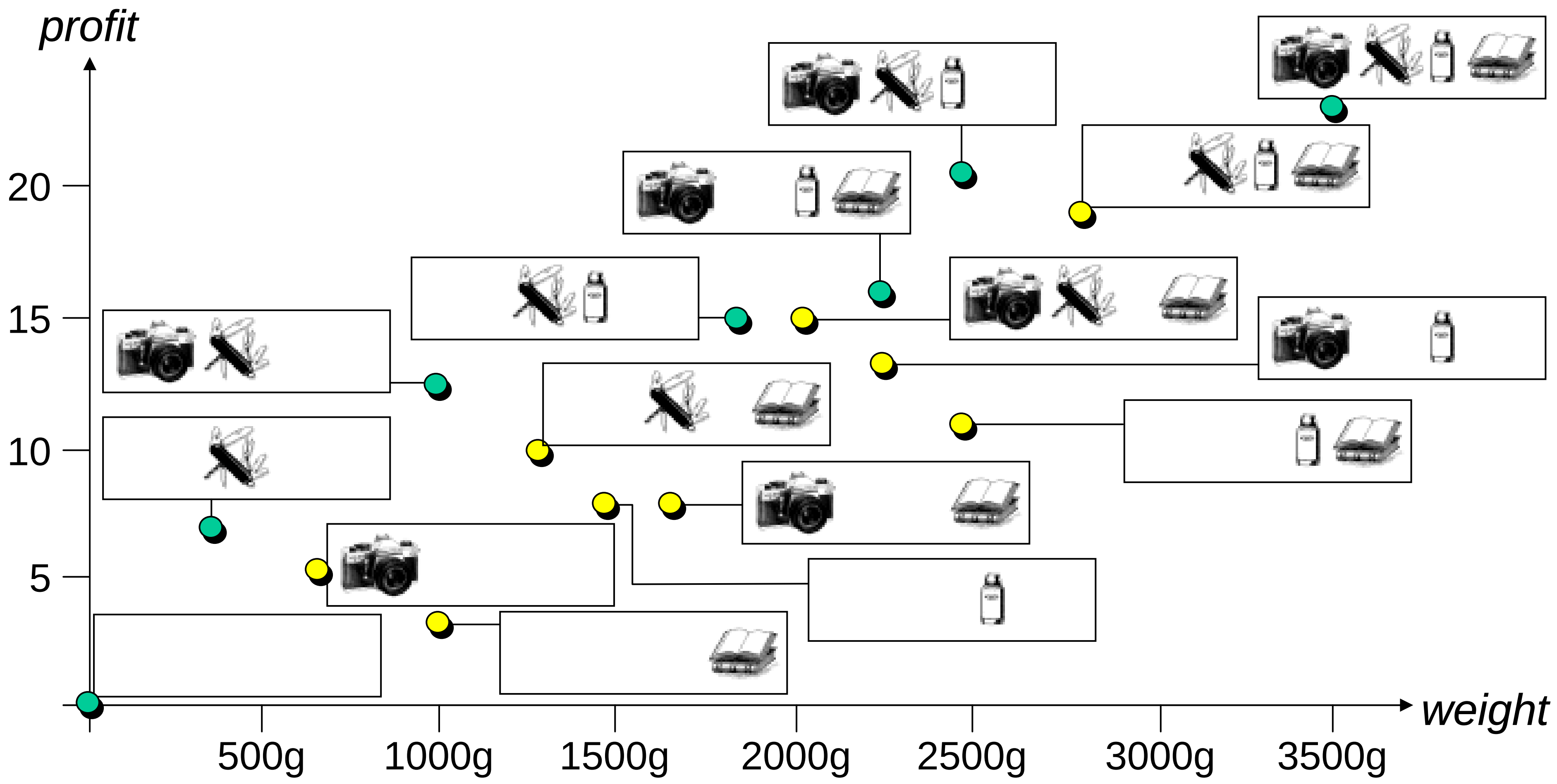


?

Goal: choose subset that

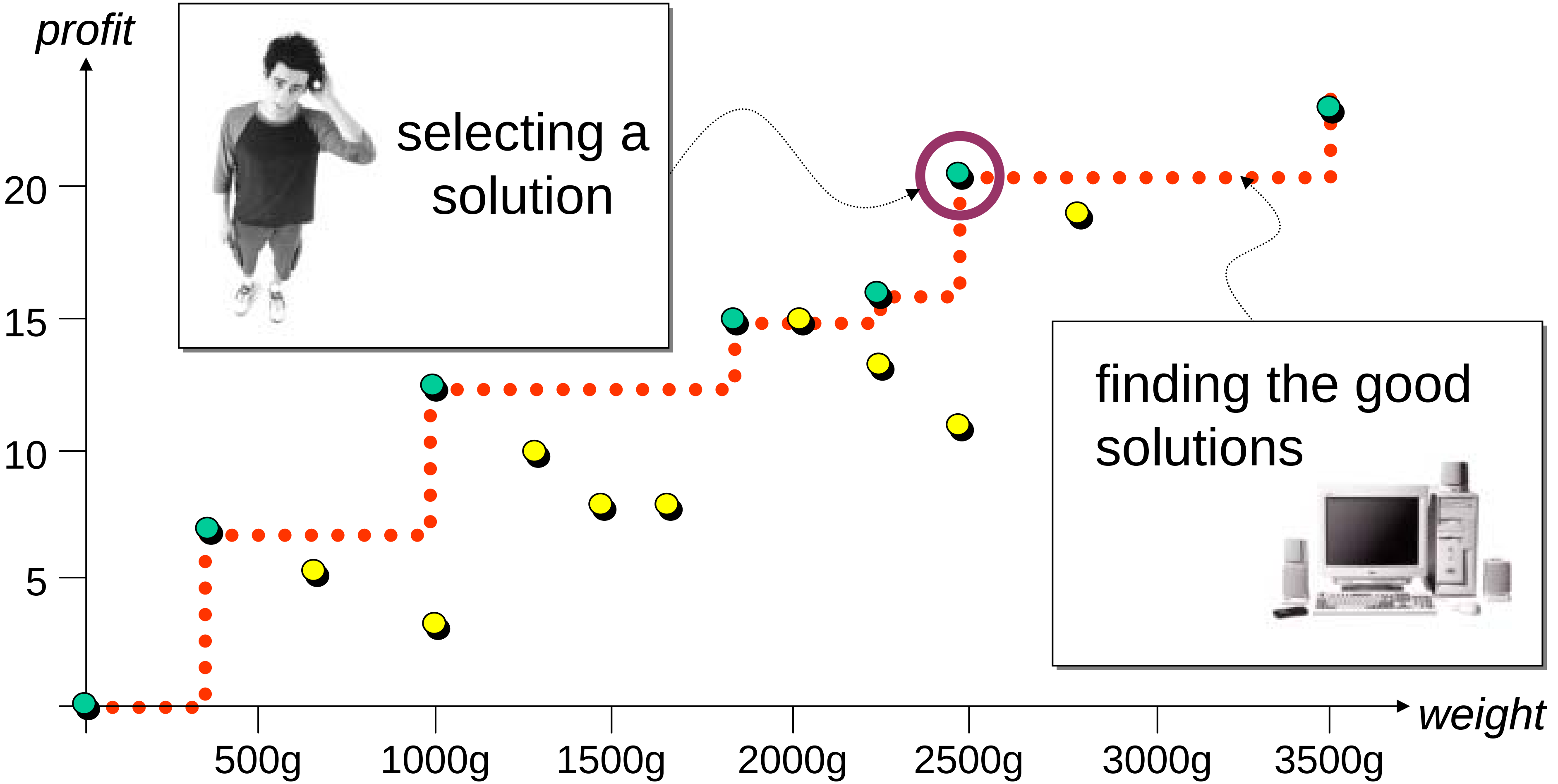
- maximizes overall profit
- minimizes total weight

The Solution Space



The Trade-off Front

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

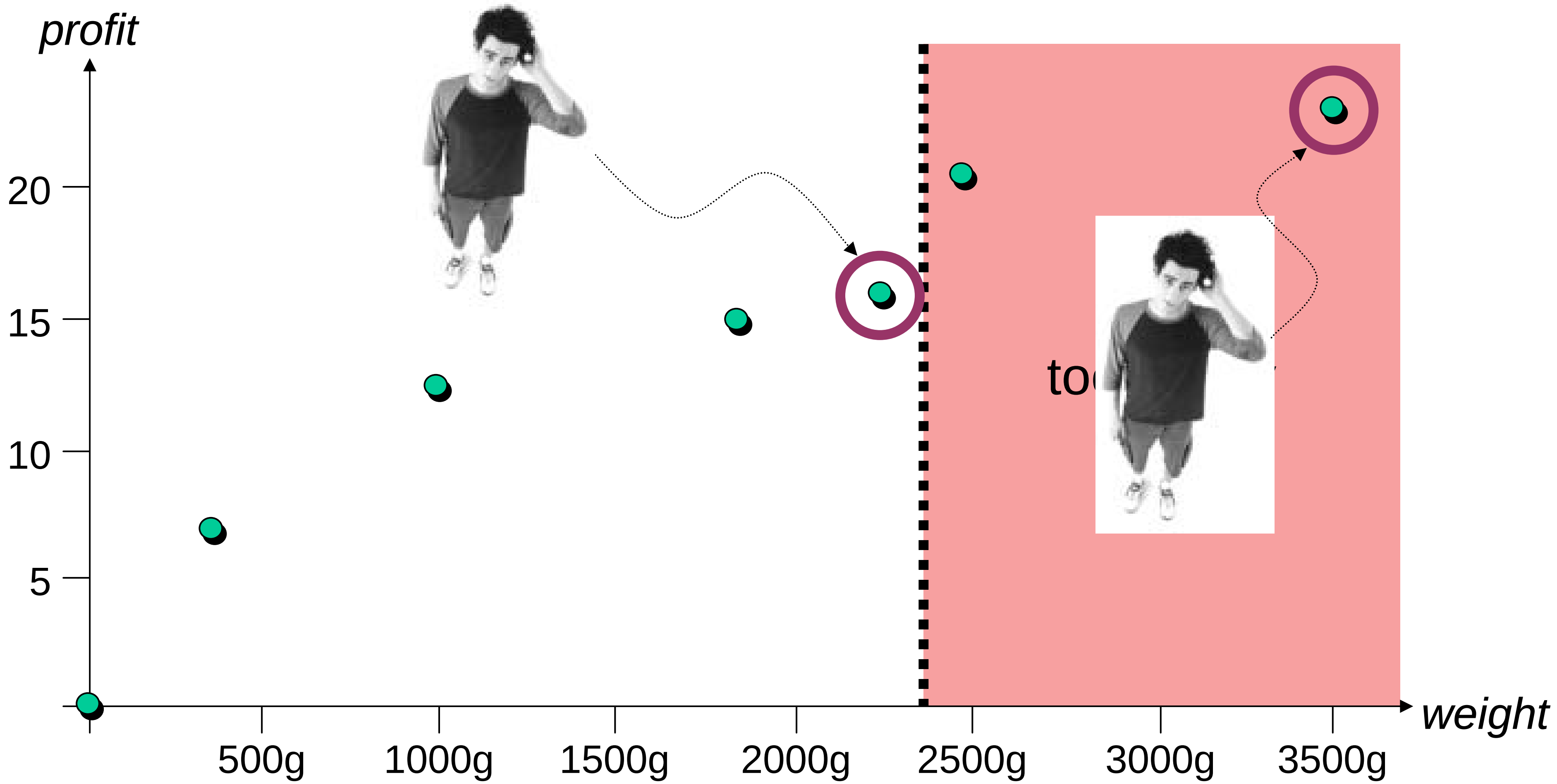


selecting a solution

finding the good solutions

Decision Making: Selecting a Solution

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



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*.

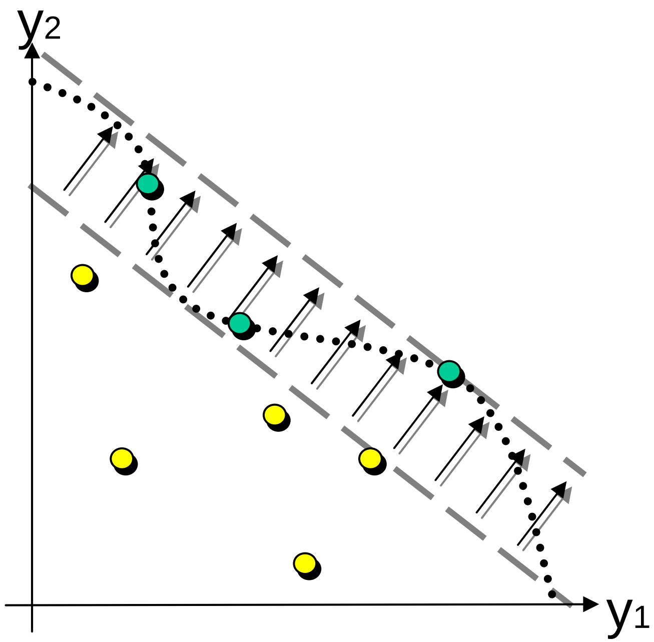
Optimization Alternatives

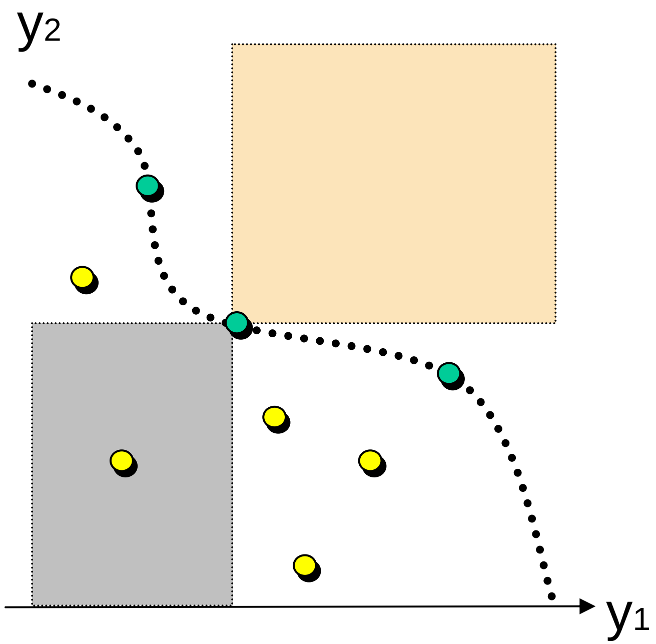
scalarization

weighted sum

population-based

SPEA2





parameter-oriented
scaling-dependent

set-oriented
scaling-independent

Scalarization Approach

parameters

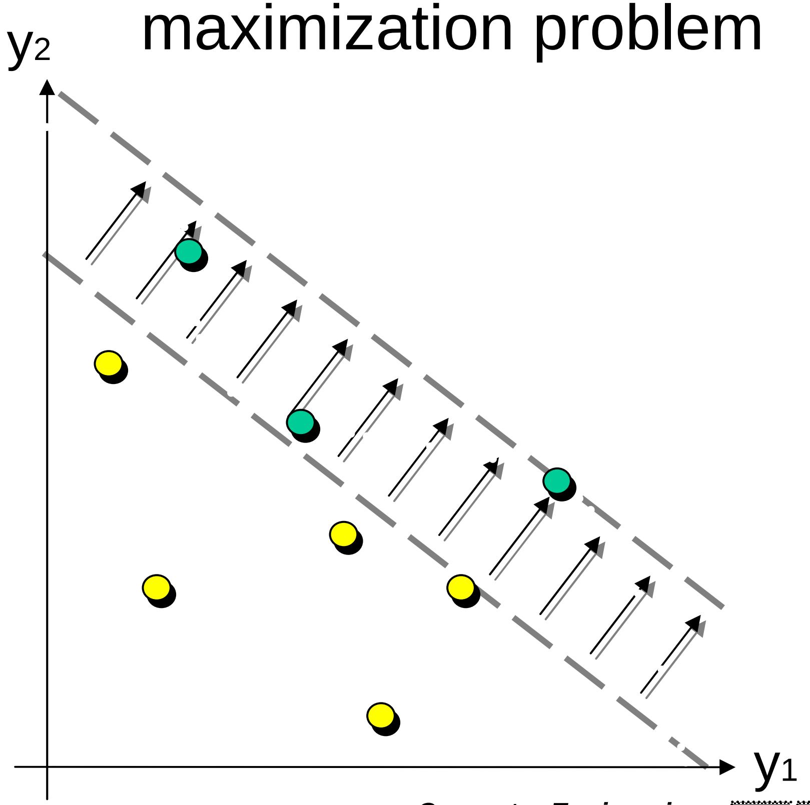
multiple objectives $(y_1, y_2, \dots, y_k)$ → transformation → single objective y

example: weighting approach

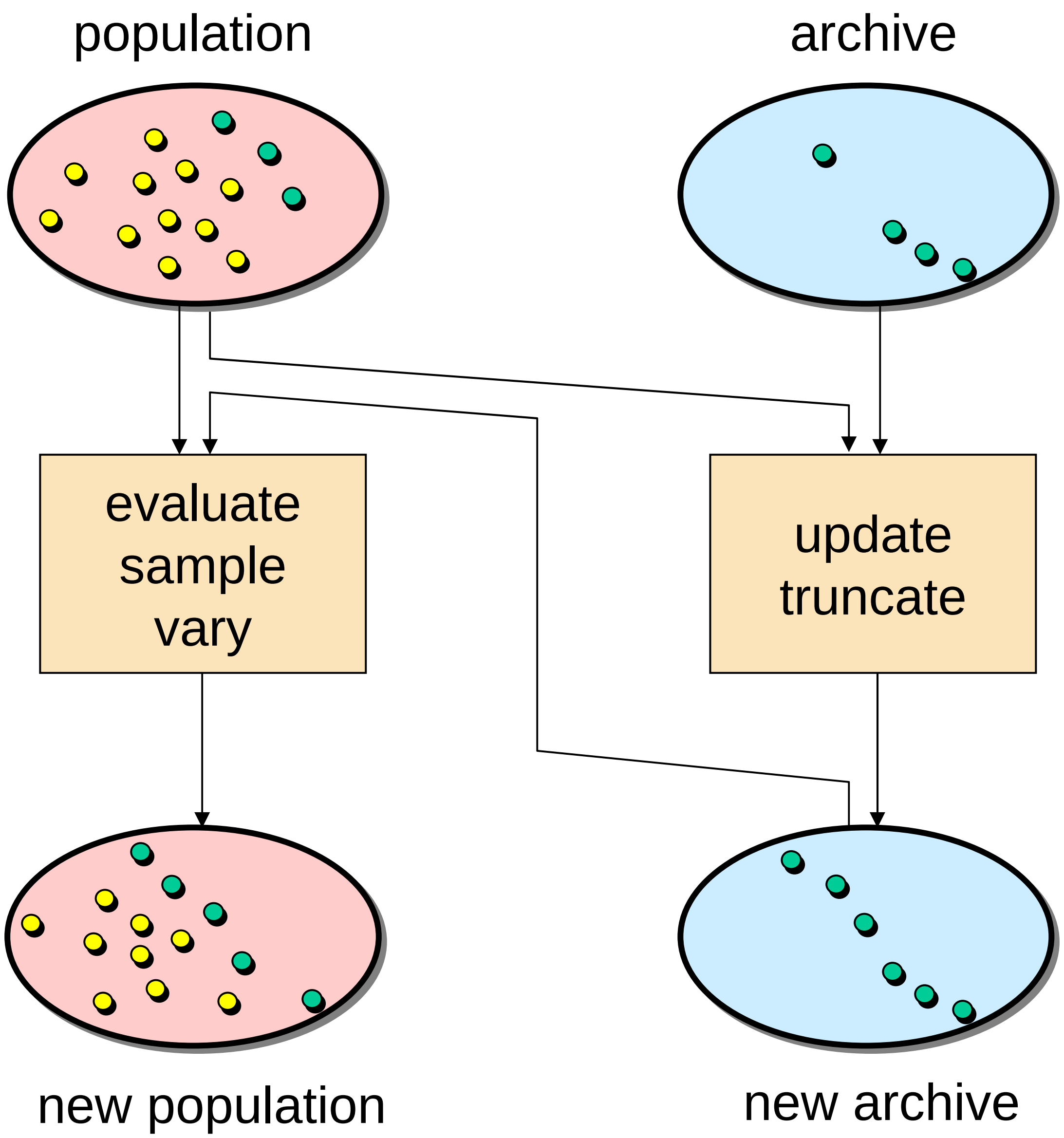
$(w_1, w_2, \dots, w_k)$

$$\rightarrow \boxed{y = w_1y_1 + \dots + w_ky_k} \rightarrow$$

maximization problem



A Generic Multiobjective EA



population

archive

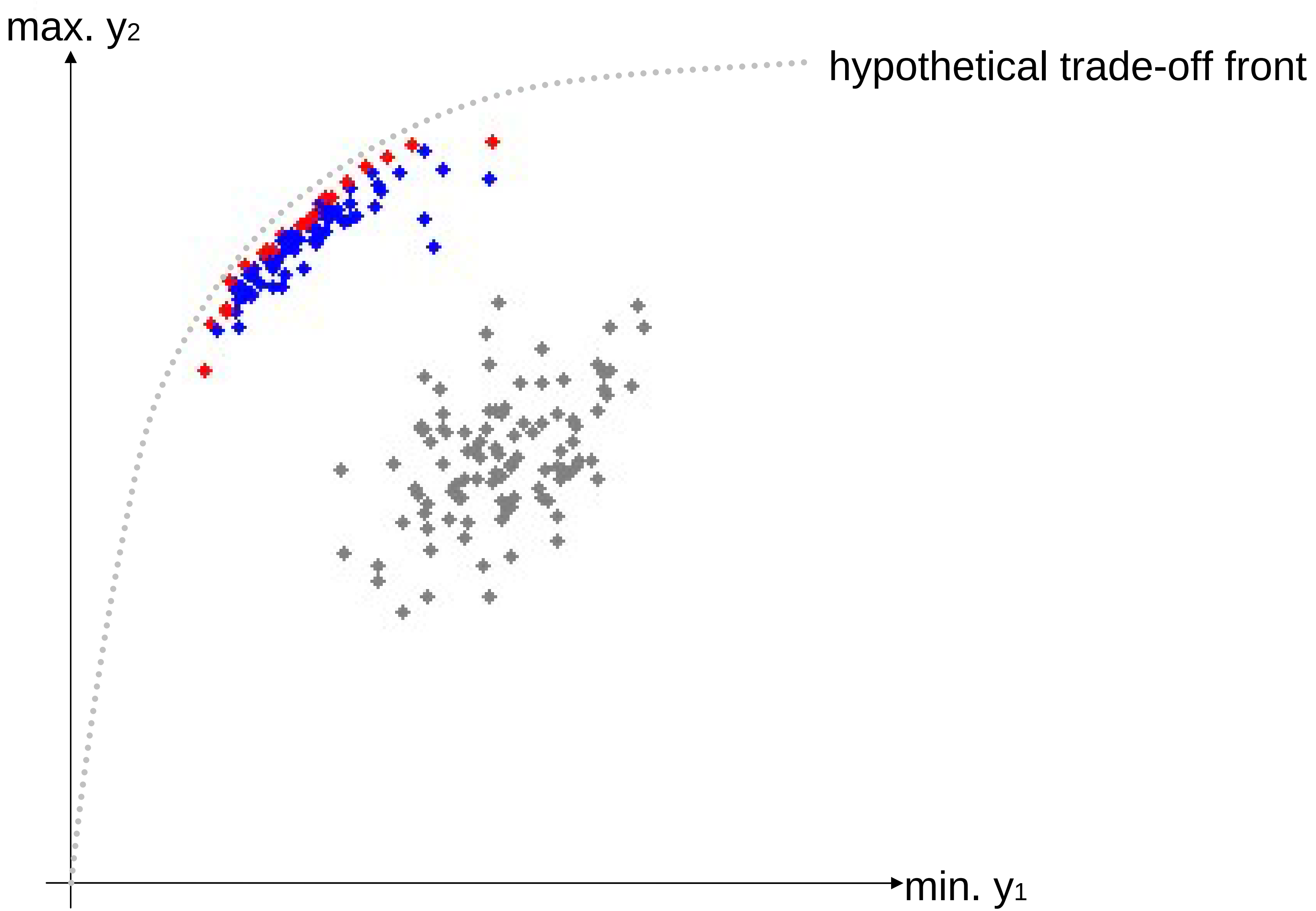
evaluate
sample
vary

update
truncate

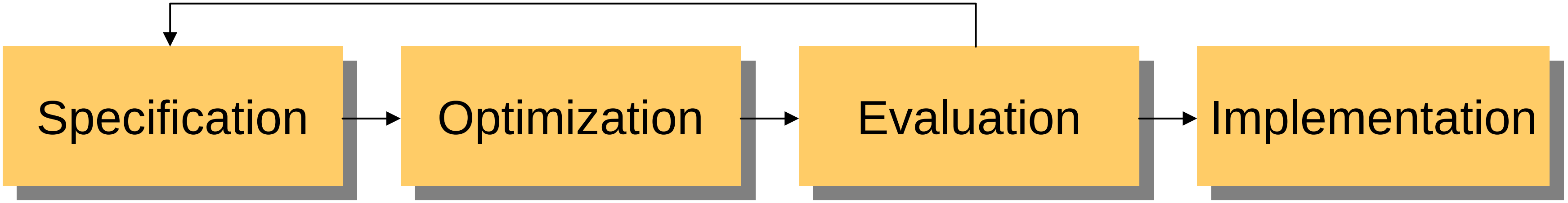
new population

new archive

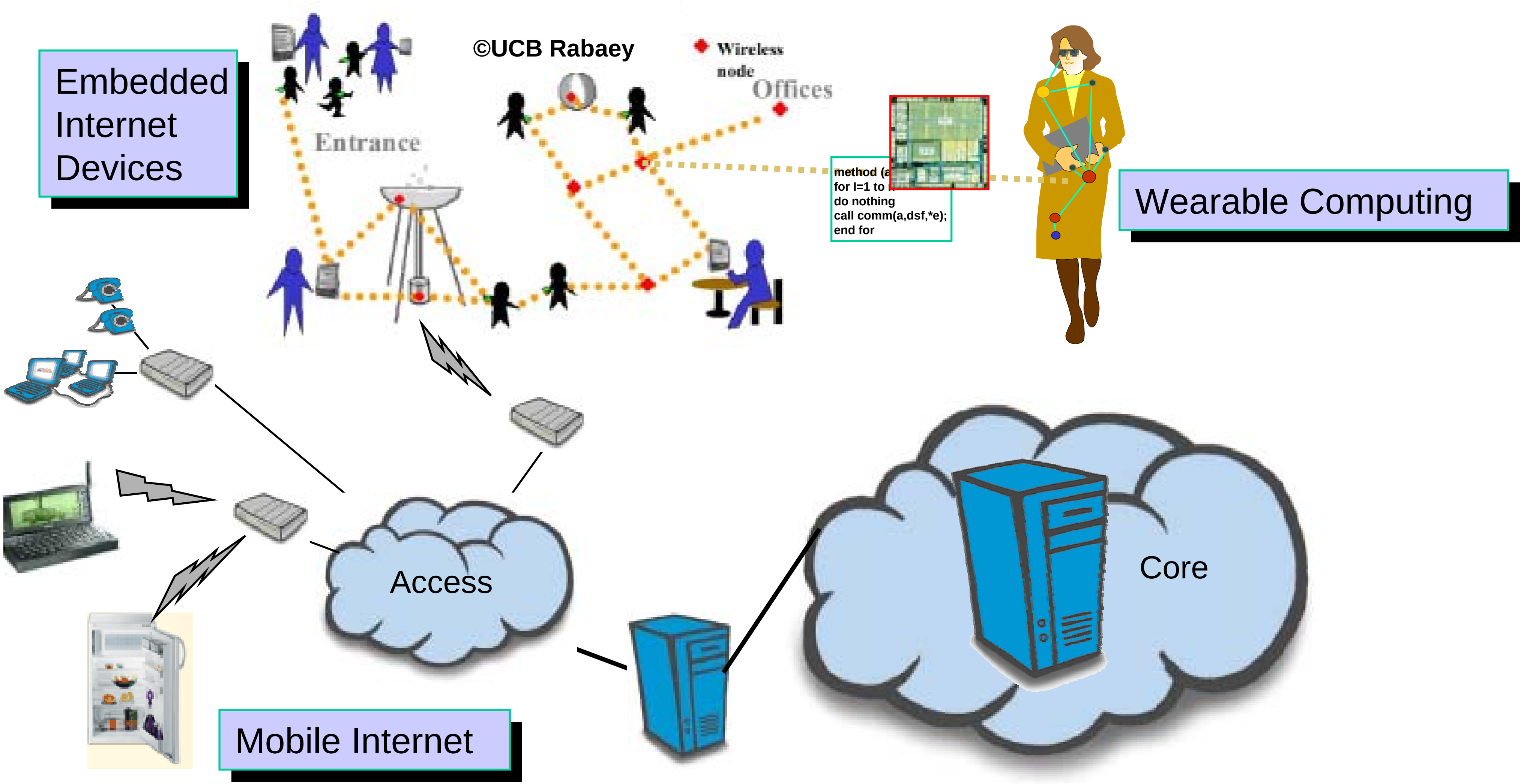
An Evolutionary Algorithm in Action



Design Space Exploration



Packet Processing in Networks



Embedded Internet Devices

©UCB Rabaey

Wearable Computing

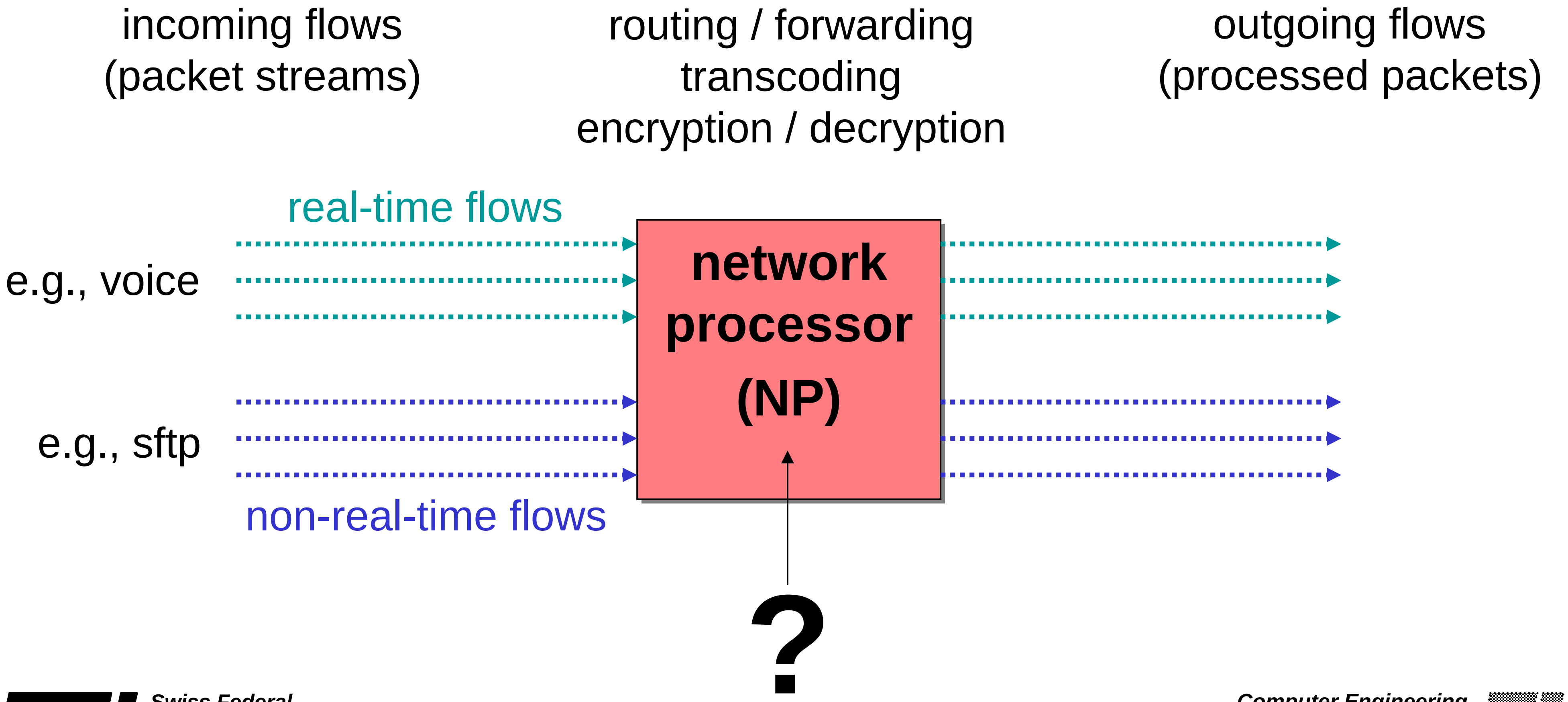
Access

Core

Mobile Internet

Network Processors

Network processor = high-performance, programmable device designed to efficiently execute communication workloads *[Crowley et al.: 2003]*



Optimization Scenario: Overview

Given:

❶ specification of the task structure (**task model**) =
for each flow the corresponding tasks to be executed

❷ different usage scenarios (**flow model**) =
sets of flows with different characteristics

Sought:

network processor implementation (**resource model**)
= architecture + task mapping + scheduling

Objectives:

❶ maximize performance

❷ minimize cost

Subject to:

❶ memory constraint

❷ delay constraints

(**performance model**)

Topics

- Multi-Objective Optimization
  - Introduction
  - ***Multi-Objective Optimization***
  - Algorithm Design
    - Design Choices
    - Representation
    - Fitness and Selection
    - Variation Operators
  - Implementation
- Design Space Exploration
  - System Design
  - Problem Specification
  - Example

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.

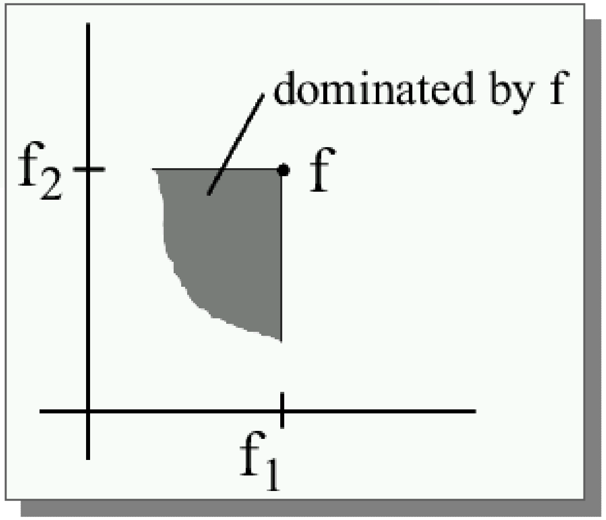
- A point is Pareto-optimal or a *Pareto-point,* if it is not dominated.

- 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. $\exists j \in \{1, \ldots, m\} : f_j > g_j$

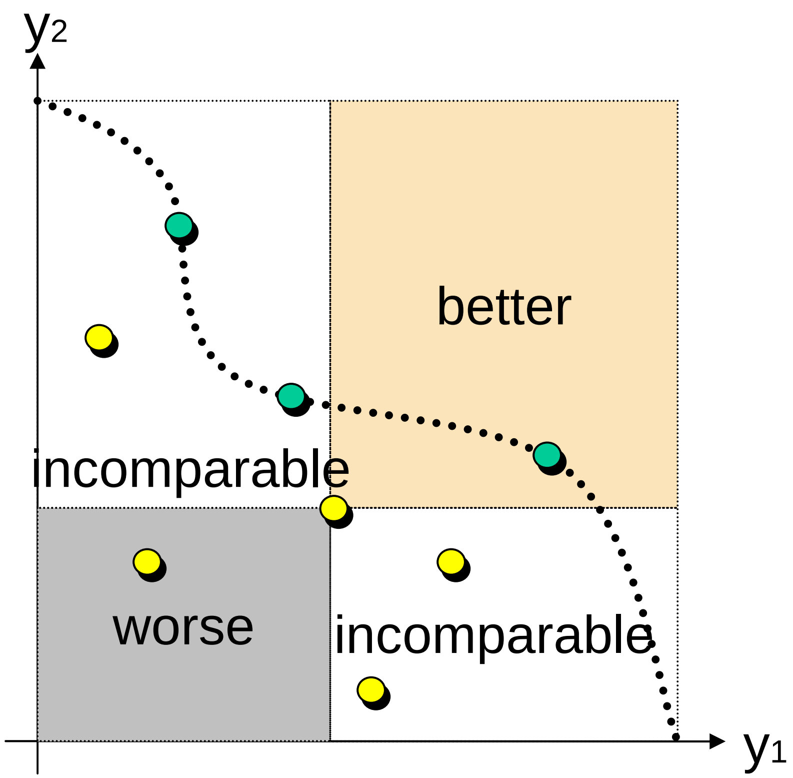


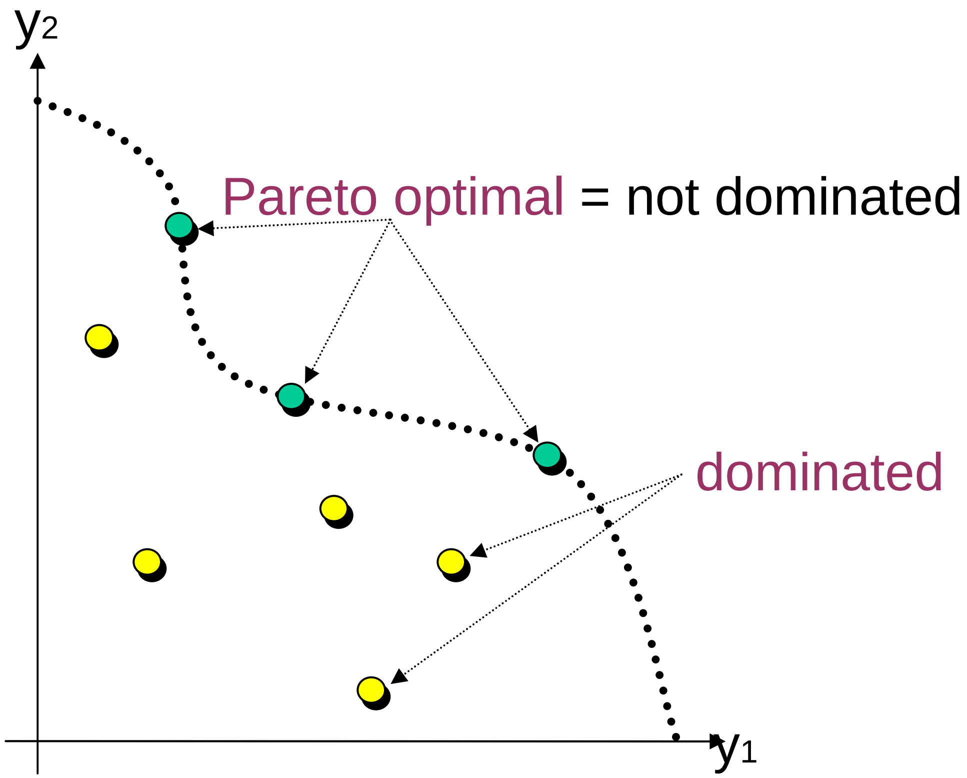
Definition 2 (Pareto set)
Let $F \subseteq \mathbb{R}^m$ be a set of vectors. Then the Pareto set $F^* \subseteq F$ is defined as follows: F^* contains all vectors $g \in F$ which are not dominated by any vector $f \in F$, i.e.

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

Multi-objective Optimization

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





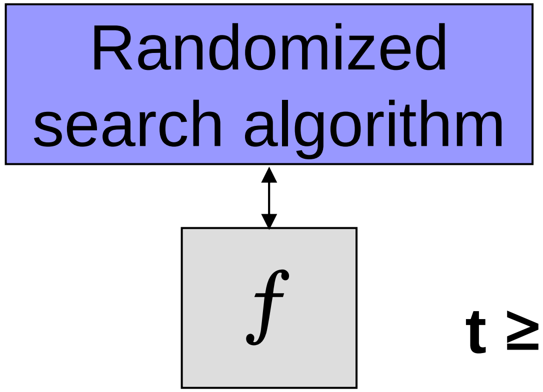
Pareto set = set of all Pareto-optimal solutions

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

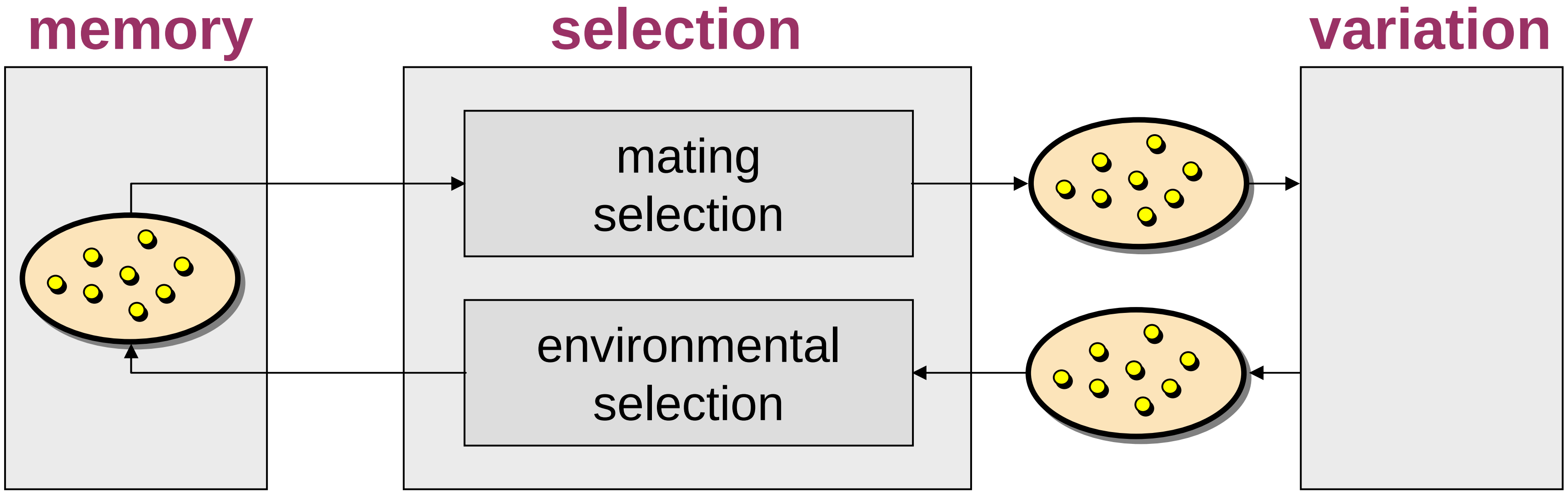
$t = 1$:

(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

Limitations of Randomized Search Algorithms

The No-Free-Lunch Theorem

All search algorithms provide in average the same performance on a all possible functions with finite search and objective spaces.

[Wolpert, McReady: 1997]

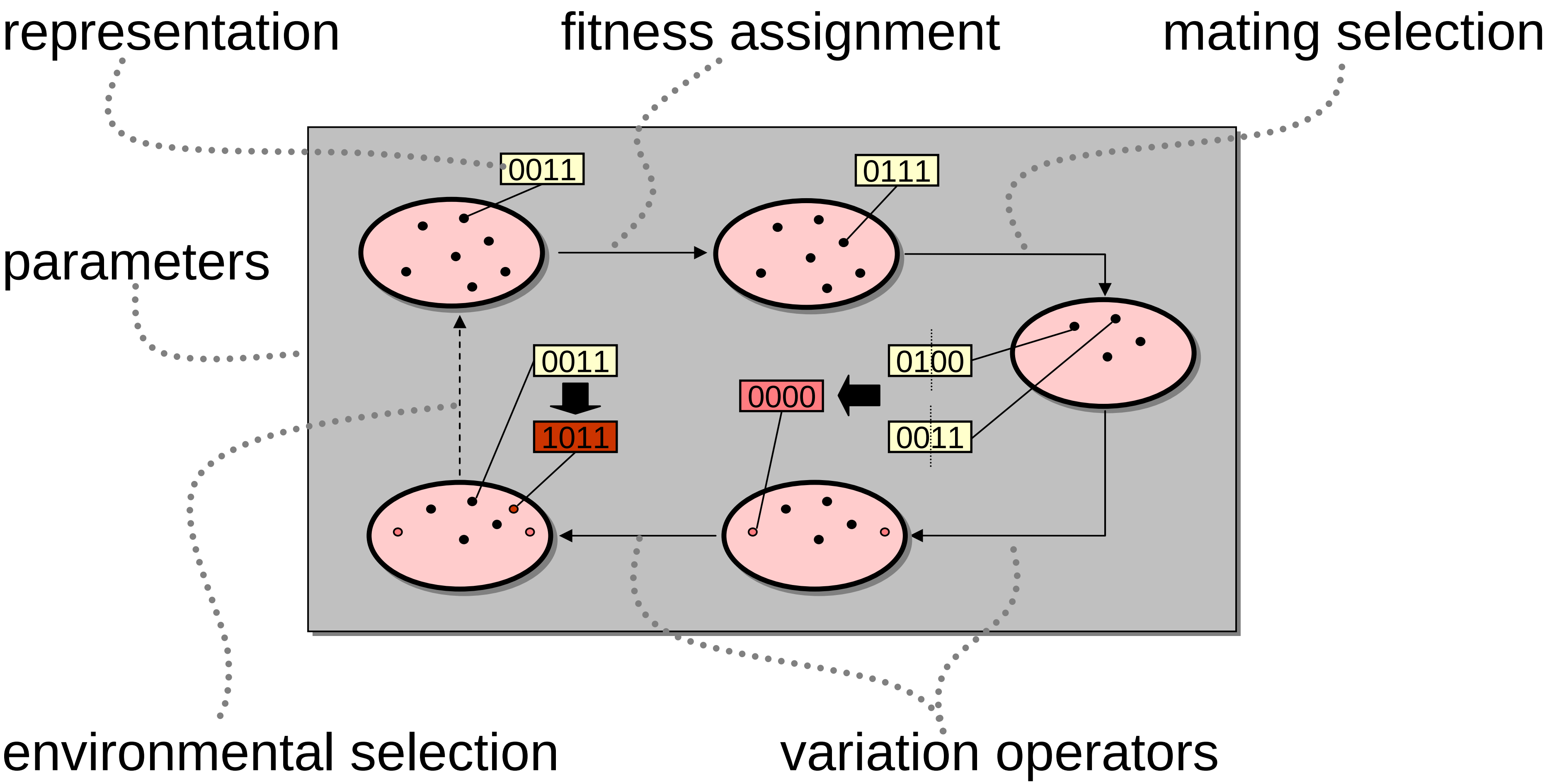
Remarks:

- Not all functions equally likely and realistic
- We cannot expect to design the algorithm beating all others
- Ongoing research: which algorithm suited for which class of problem?

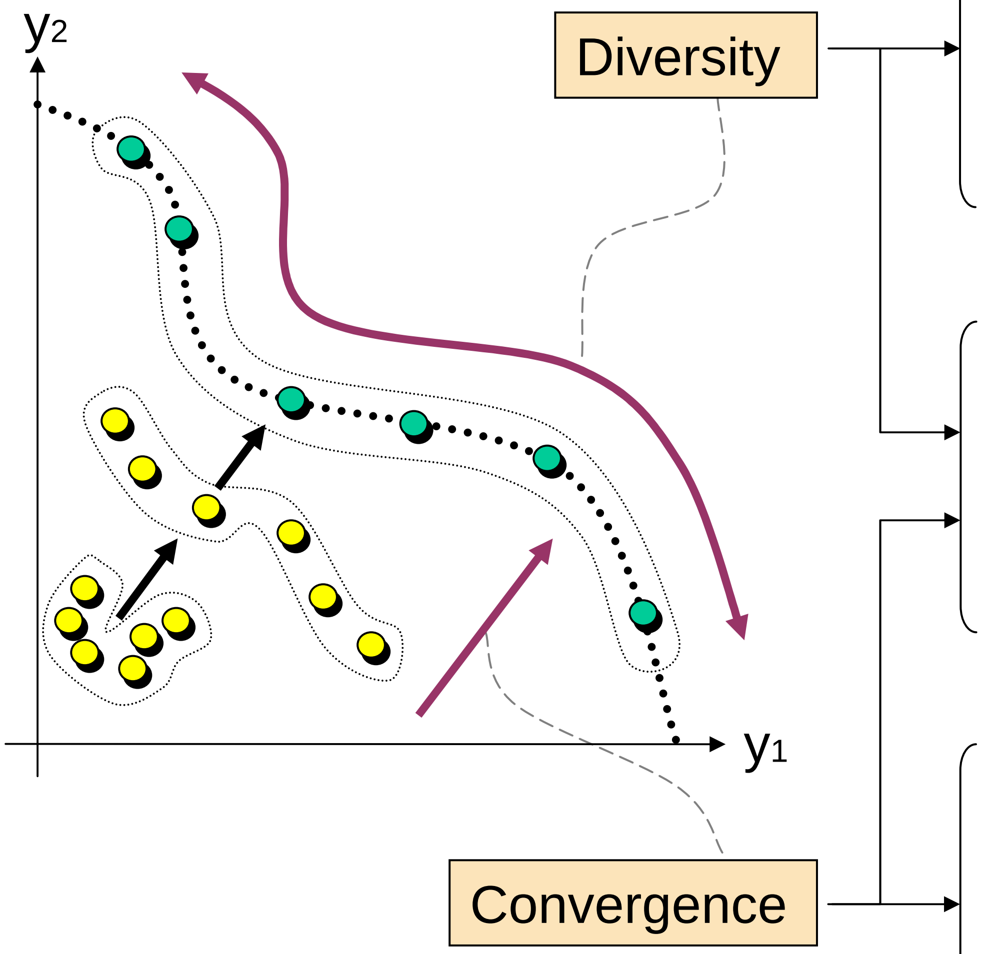
Topics

- Multi-Objective Optimization
  - Introduction
  - Multi-Objective Optimization
  - ***Algorithm Design***
    - ***Design Choices***
    - Representation
    - Fitness and Selection
    - Variation Operators
  - Implementation
- Design Space Exploration
  - System Design
  - Problem Specification
  - Example

Design Choices



Issues in Multi-Objective Optimization



- How to maintain a diverse Pareto set approximation?

  ❷ **density estimation**

- How to prevent nondominated solutions from being lost?

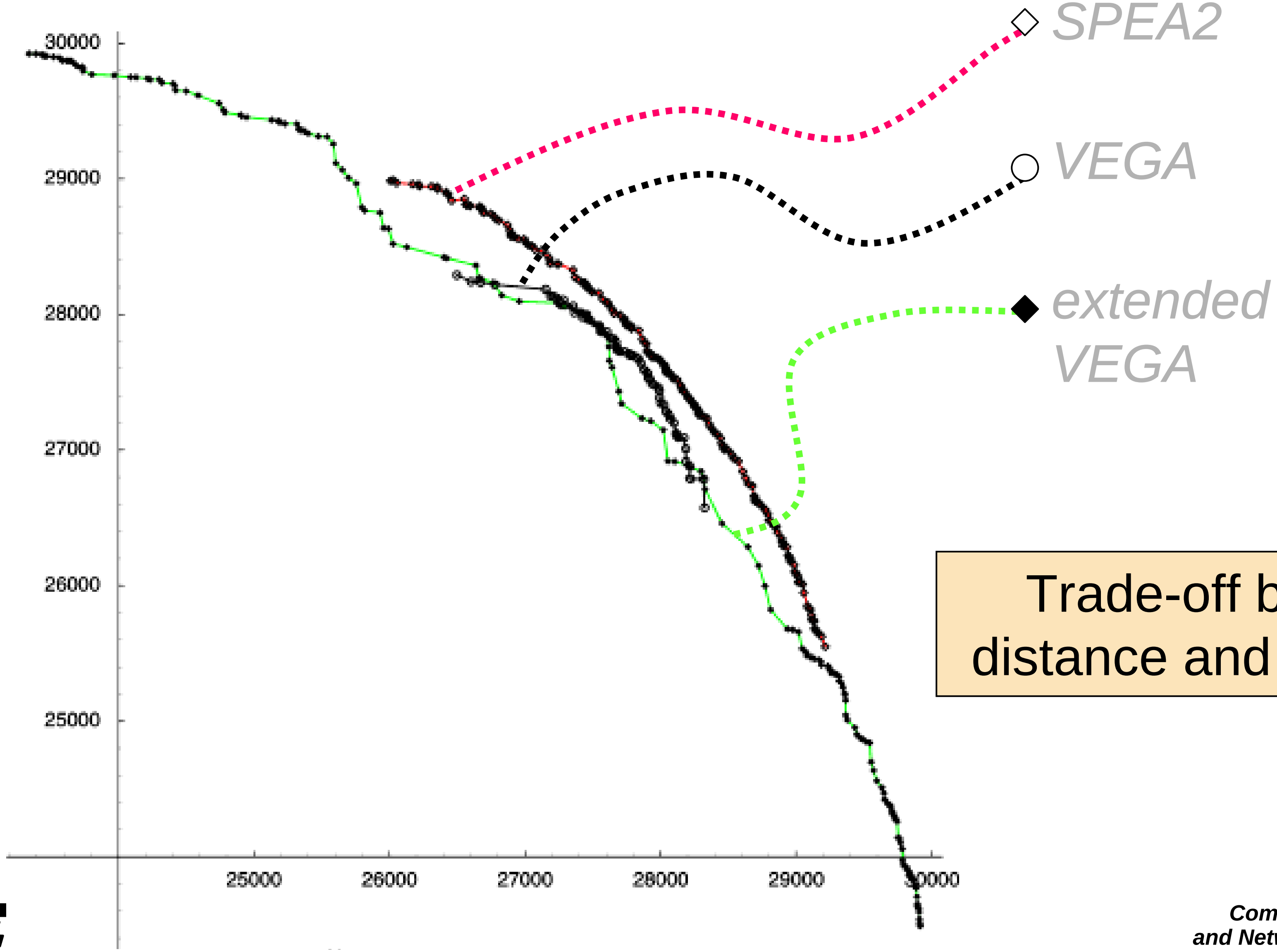
  ❸ **environmental selection**

- How to guide the population towards the Pareto set?

  ❶ **fitness assignment**

Comparison of Three Implementations

2-objective knapsack problem

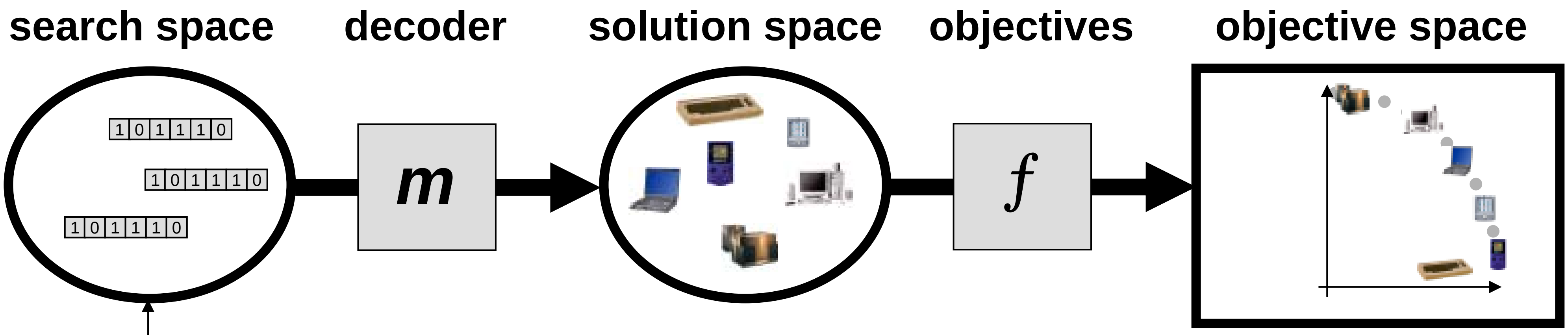


Trade-off between distance and diversity?

Topics

- Multi-Objective Optimization
  - Introduction
  - Multi-Objective Optimization
  - Algorithm Design
    - Design Choices
    - ***Representation***
    - Fitness
    - Variation Operators
  - Implementation
- Design Space Exploration
  - System Design
  - Problem Specification
  - Example

Representation



search space **decoder** **solution space** **objectives** **objective space**

solutions encoded by vectors, matrices, trees, lists, ...

fixed length variable length

Issues:

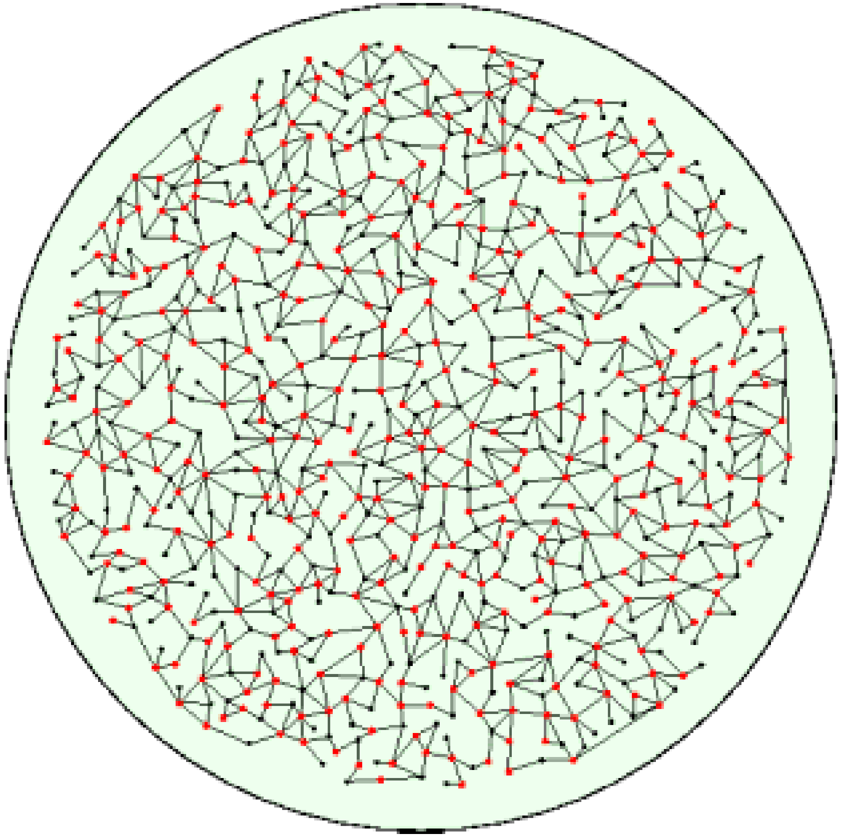
- completeness (each solution has an encoding)
- uniformity (all solutions are represented equally)
- redundancy (cardinality of search space vs. solution space)
- feasibility (each encoding maps to a feasible solution)

Example: Binary Vector Encoding

Given: graph

Goal: find minimum subset of nodes such that each edge is connected to at least one node of this subset (minimum vertex cover)

nodes	A	B	C	D	E	F	…
selected?	1	0	1	1	1	0	…

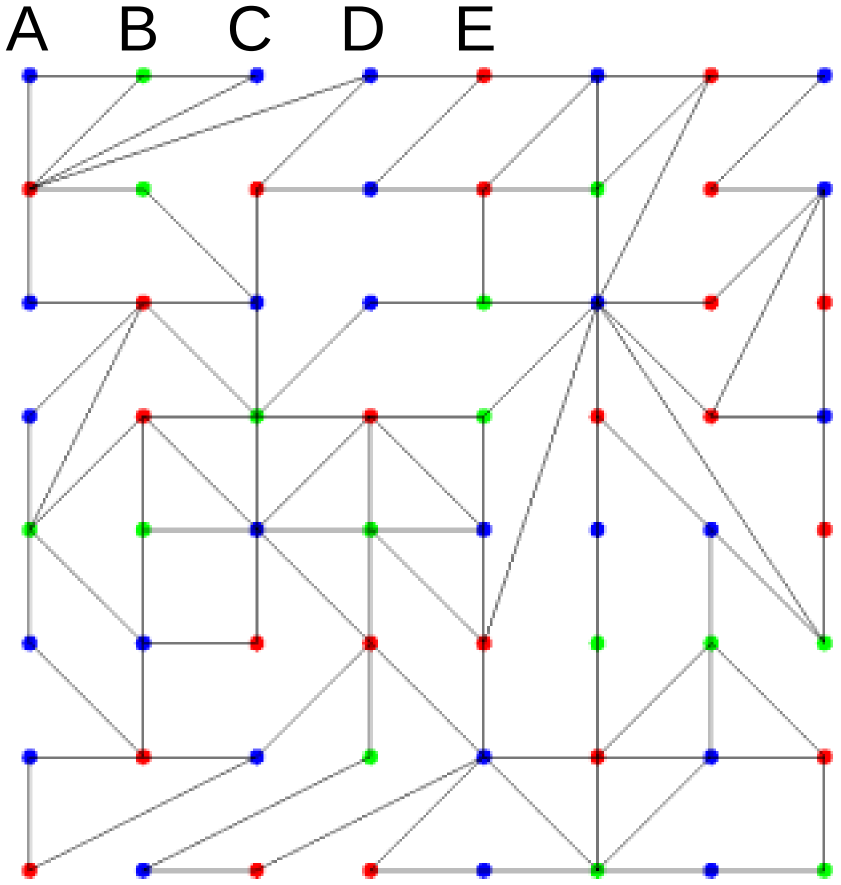


Example: Integer Vector Encoding

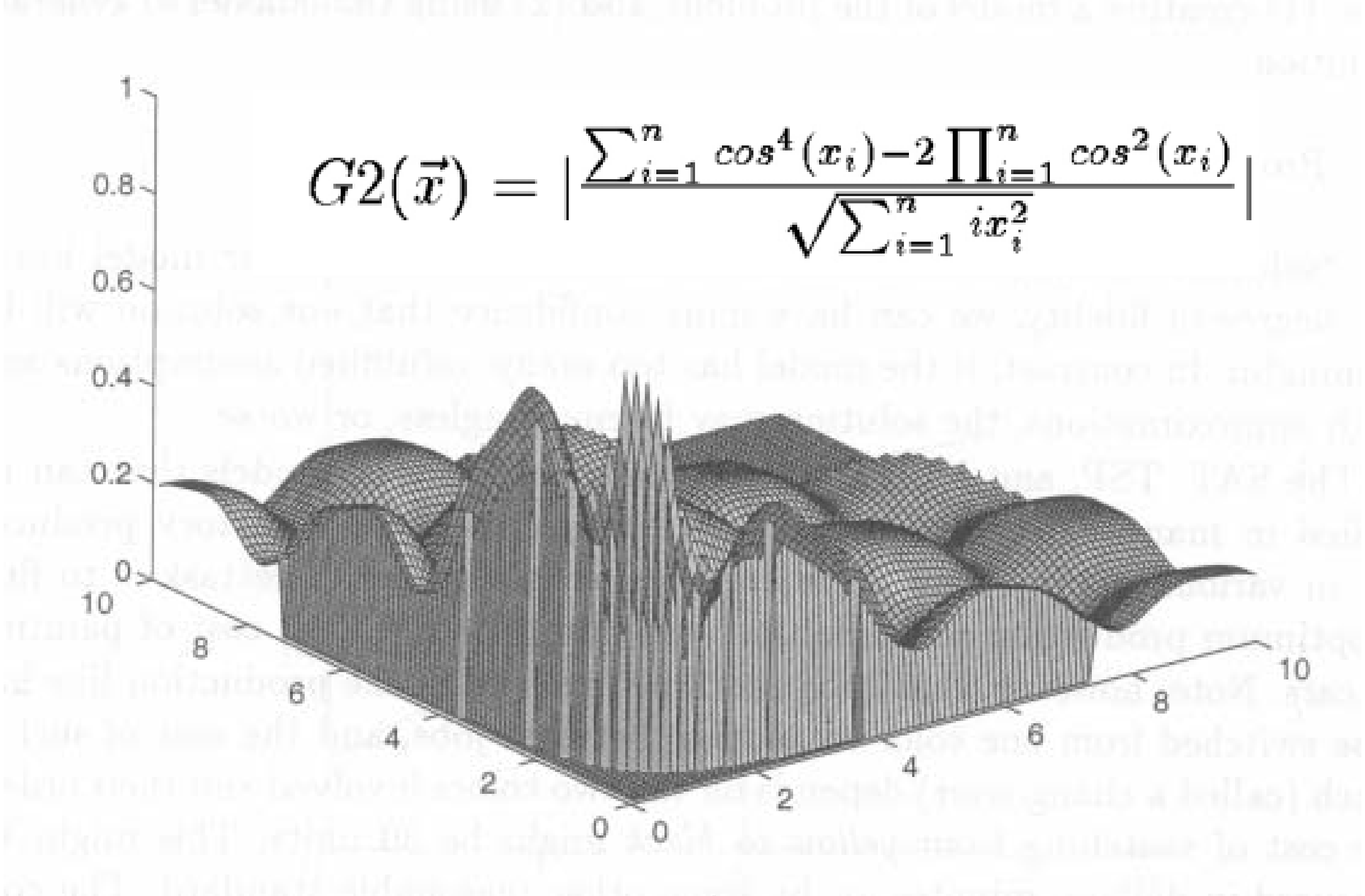
Given: graph, k colors

Goal: assign each node one of the k colors such that the number of connected nodes with the same color is minimized (graph coloring problem)

nodes	A	B	C	D	E	F	…
colors	1	2	1	3	1	2	…



Example: Real Vector Encoding

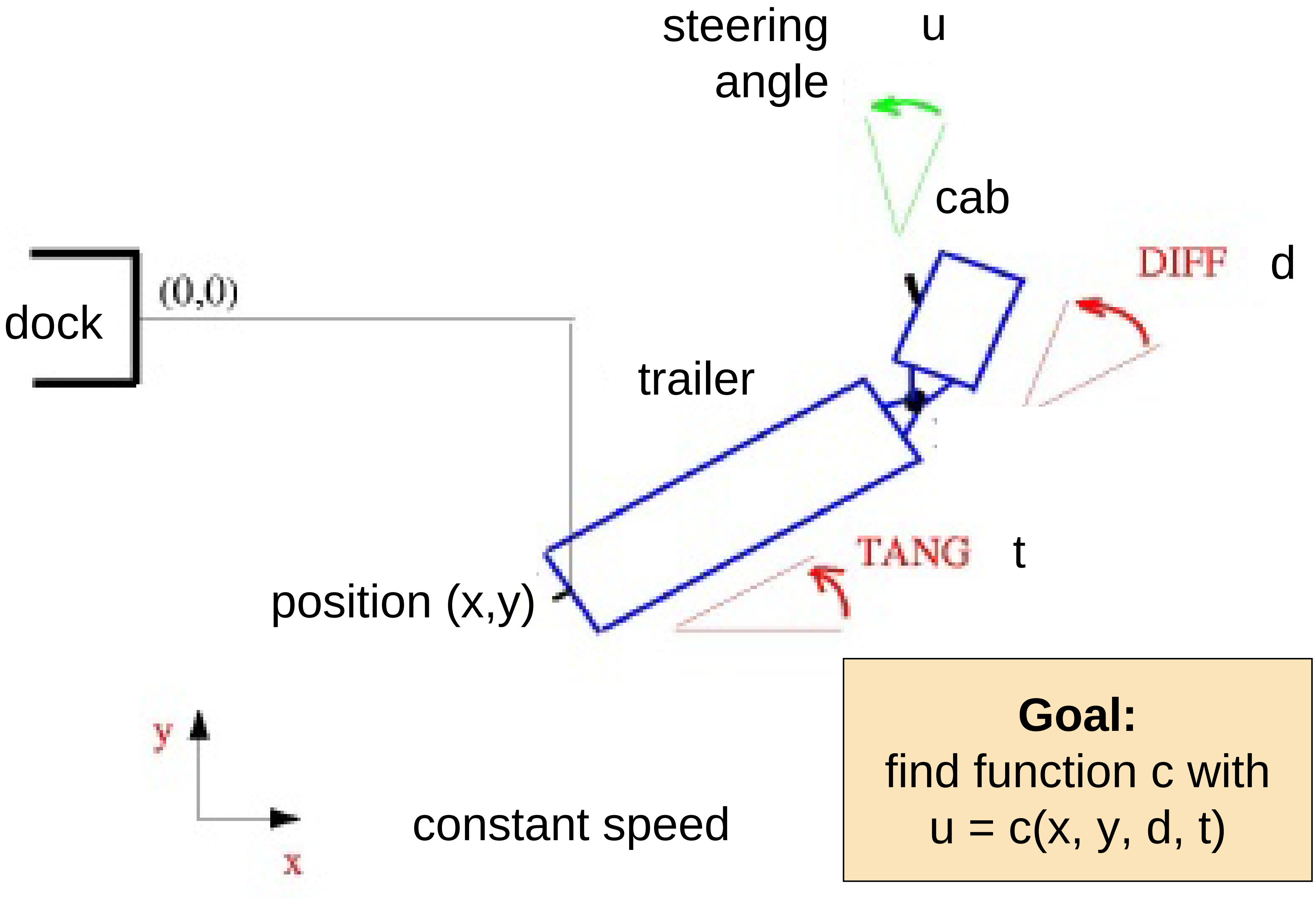


$$G2(\vec{x}) = \left| \frac{\sum_{i=1}^{n} cos^4(x_i) - 2\prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$$

parameters	x_1	x_2	x_3	x_4	…	x_n
values	0.33	0.53	1.03	3.25	…	9.83

[Michalewicz, Fogel: How to Solve it. Springer 2000]

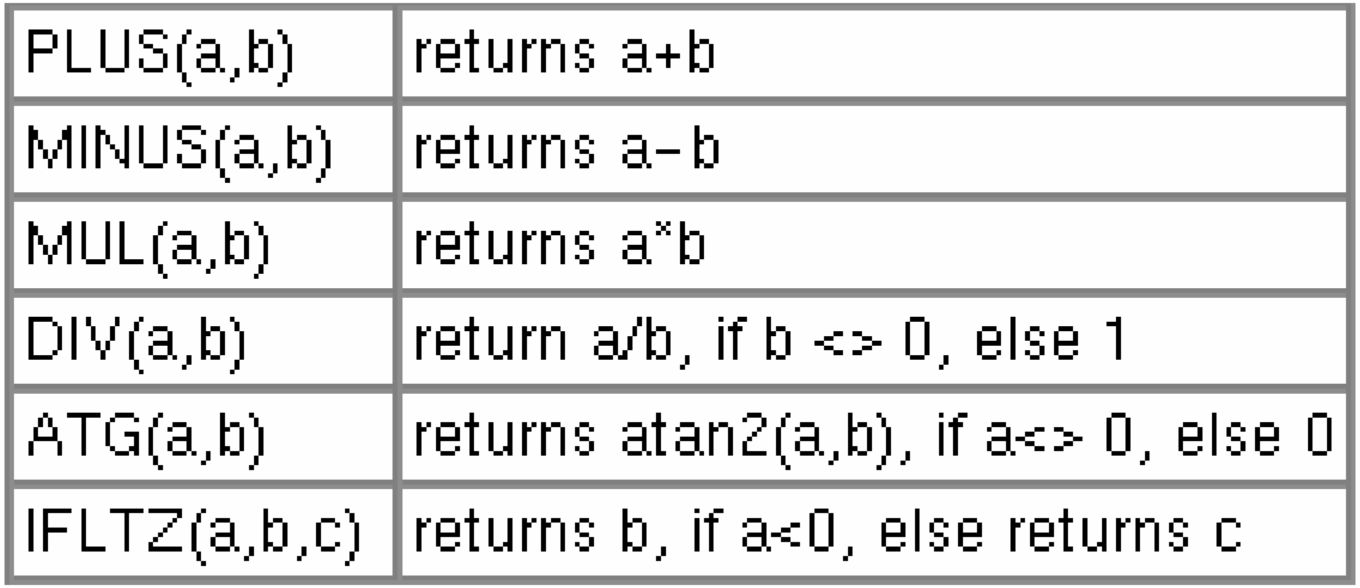
Tree Example: Parking a Truck



Goal:
find function c with
u = c(x, y, d, t)

Search Space for the Truck Problem

Operators:



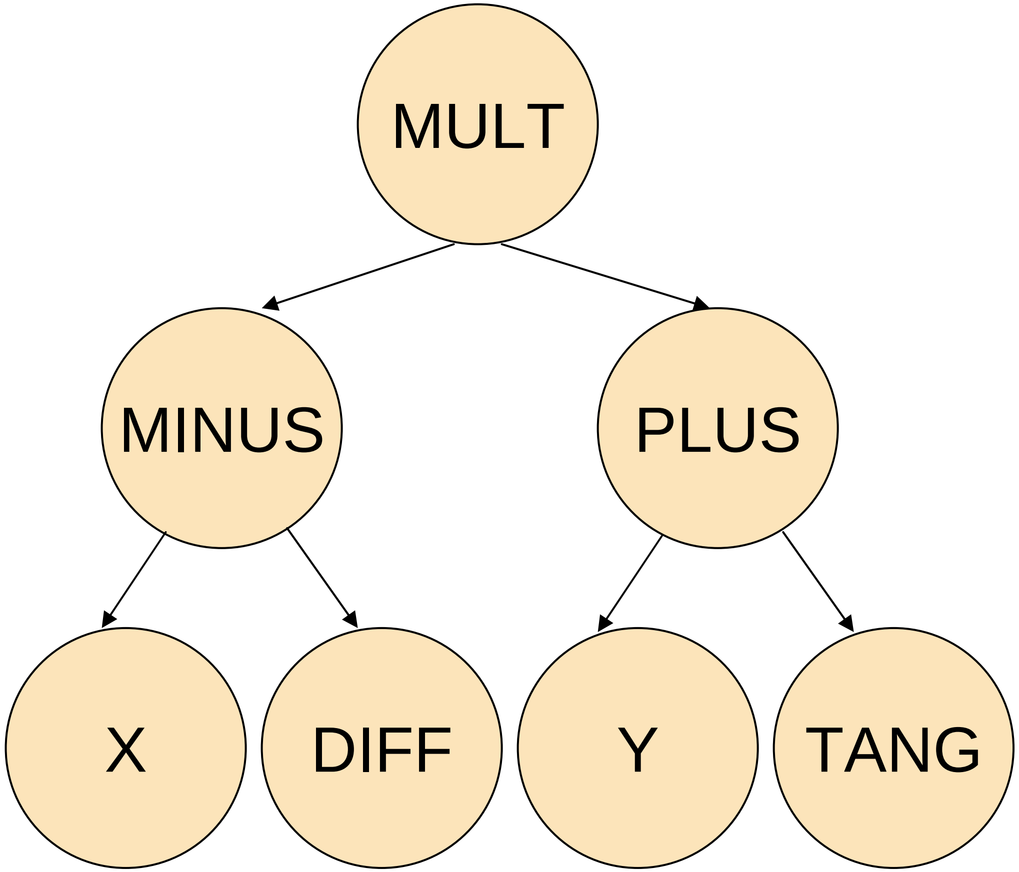
PLUS(a,b)	returns a+b
MINUS(a,b)	returns a-b
MUL(a,b)	returns a*b
DIV(a,b)	return a/b, if b <> 0, else 1
ATG(a,b)	returns atan2(a,b), if a<> 0, else 0
IFLTZ(a,b,c)	returns b, if a<0, else returns c

Arguments:

X	position x
Y	position y
DIFF	cab angle d
TANG	trailer angle t

Search space : set of symbolic expression using the above operators and arguments

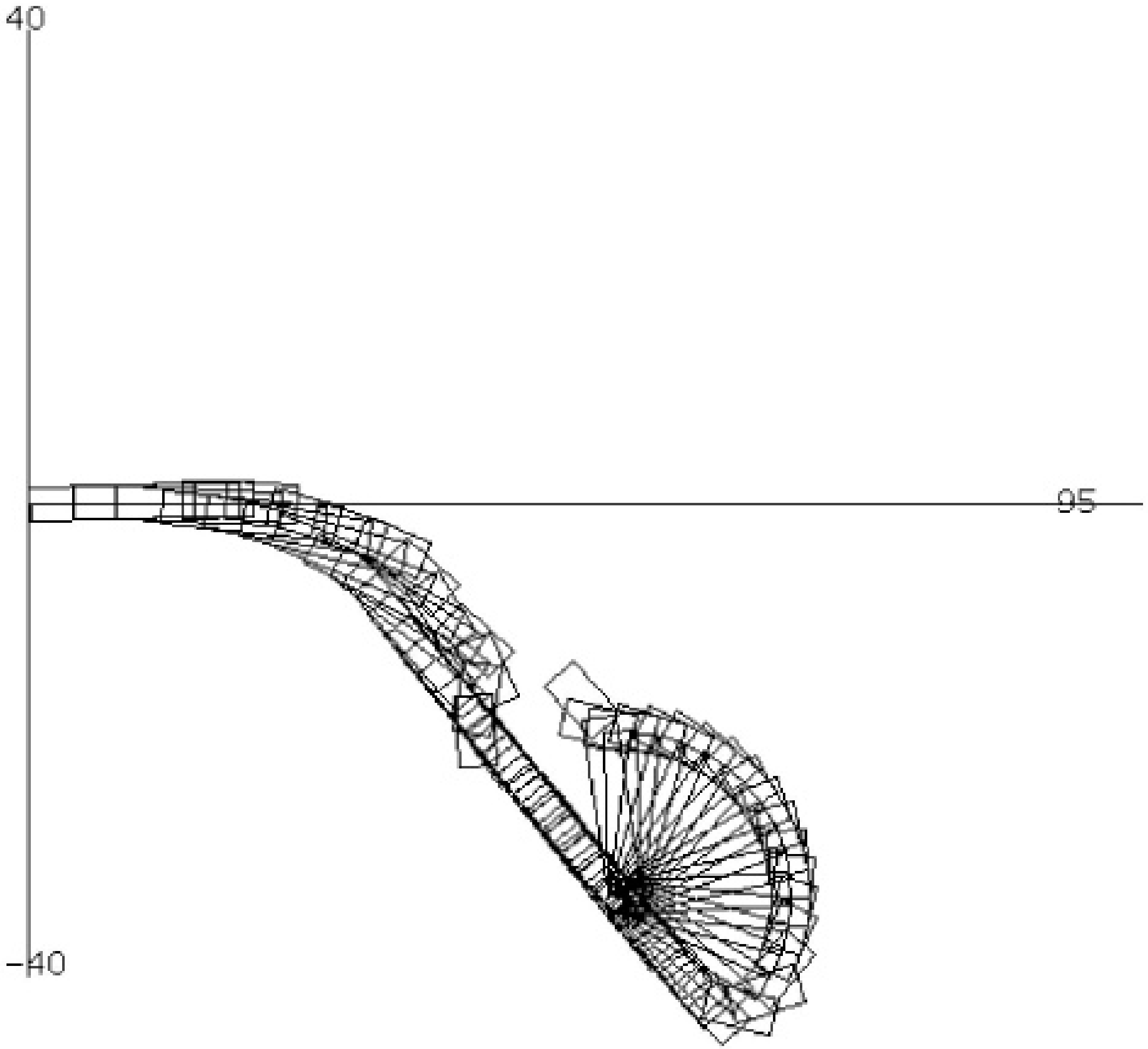
Example Solution: Tree Representation



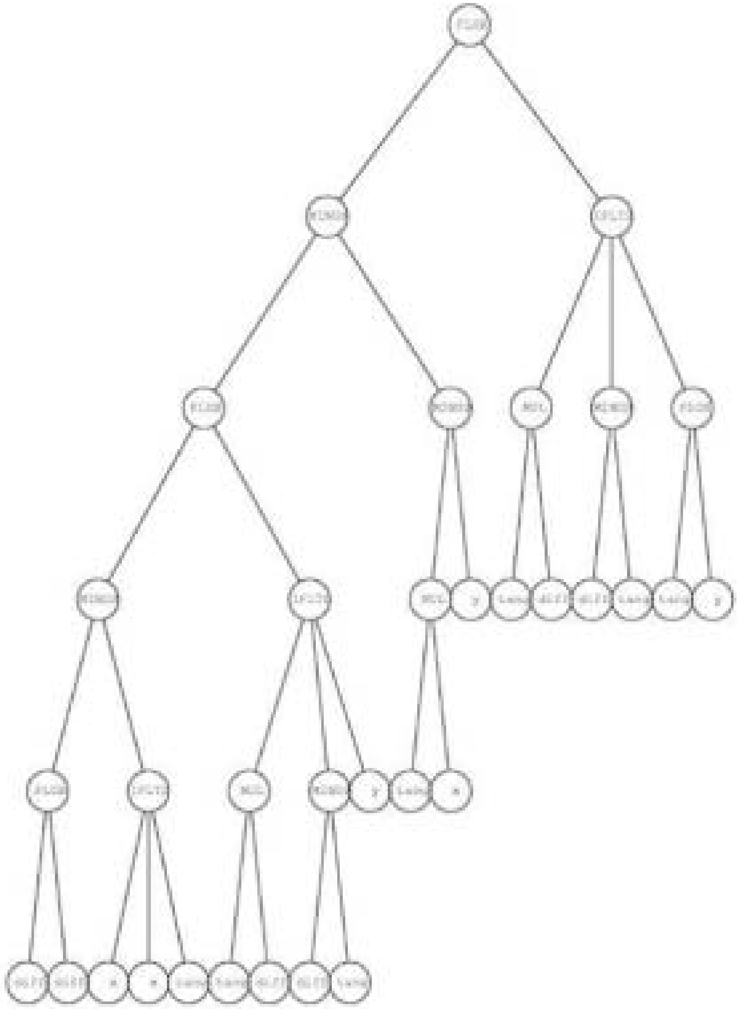
encodes the function (symbolic expression): $u = (x - d) * (y + t)$

A Solution Found by an EA

truck simulation



encoded tree



Topics

- Multi-Objective Optimization
  - Introduction
  - Multi-Objective Optimization
  - Algorithm Design
    - Design Choices
    - Representation
    - ***Fitness and Selection***
    - Variation Operators
  - Implementation
- Design Space Exploration
  - System Design
  - Problem Specification
  - Example

Fitness Assignment

Fitness = scalar value representing quality of an individual (usually). Used for mating and environmental selection

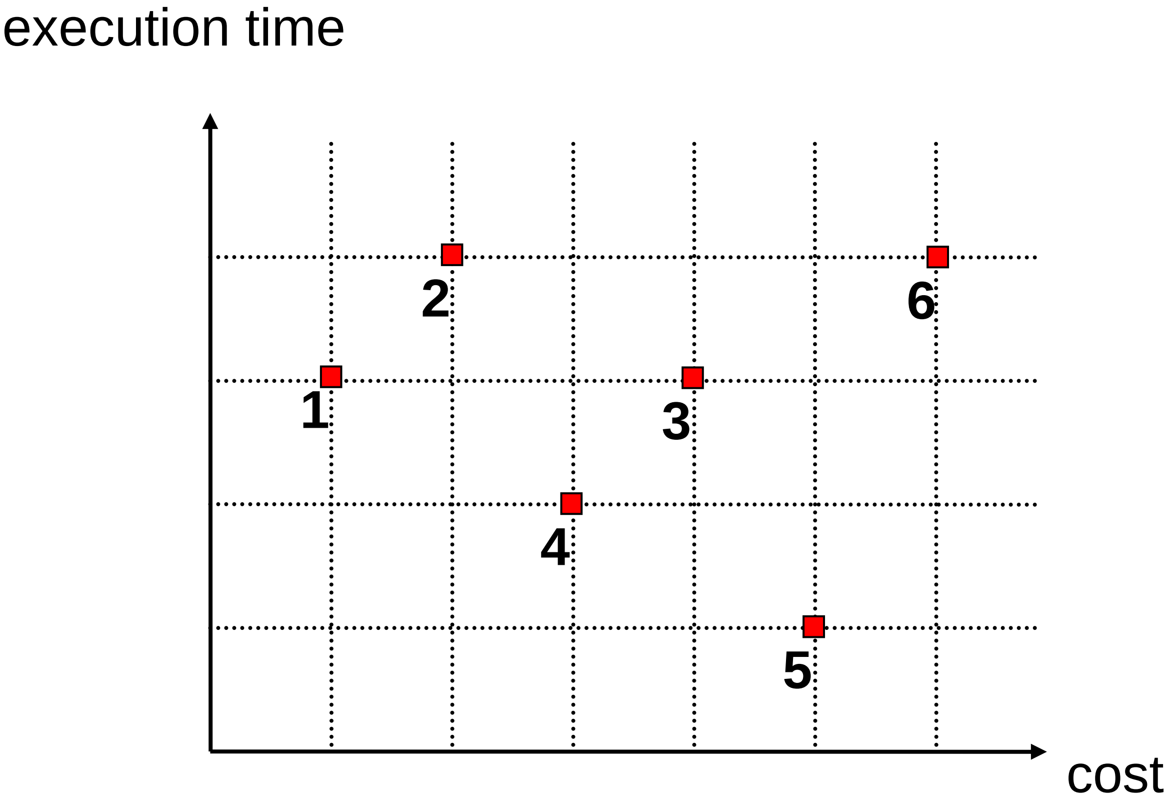
$$F_i = f\,(\,m\,(\mathrm{i})\,)$$

Difficulties:

- multiple objectives have to be considered (Pareto set is sought)
- multiple optima need to be approximated (how to consider diversity?)
- constraints are involved which have to be met

Example Pareto Ranking

- ***Fitness function***: $F(J) = \sum_{i=1,\ldots,N,J\neq J_i} \begin{cases} 1 : J_i \prec J \\ 0 : \text{else} \end{cases}$



$$F(1) = 3$$

$$F(2) = 1$$

$$F(3) = 1$$

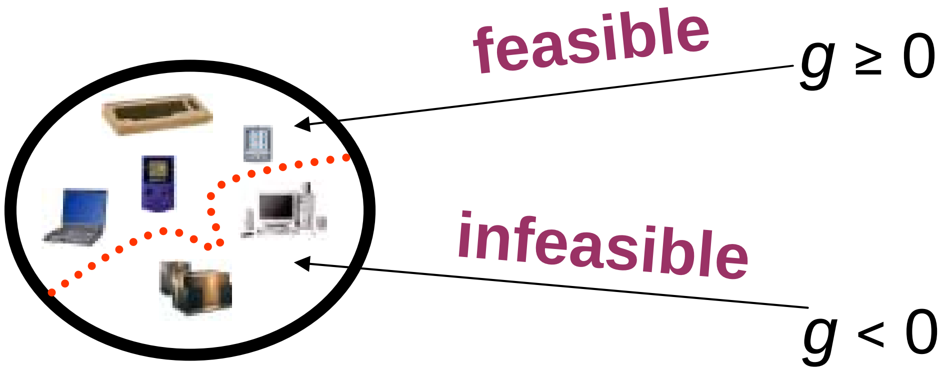
$$F(4) = 2$$

$$F(5) = 1$$

$$F(6) = 0$$

Constraint Handling

Constraint = $g(x_1, x_2, \ldots, x_n) \geq 0$



Approaches:

- construct initialization and variation such that infeasible solutions are not generated (resp. not inserted)
- representation is such that decoding always yields a feasible solution
- calculate constraint violation ($- g(x_1, x_2, \ldots, x_n)$) and incorporate it into fitness, e.g., $F_i = f\,(\,m\,(i)\,) - g(x_1, x_2, \ldots, x_n)$ (fitness to be maximized)
- code constraint as a new objective

Selection

Two conflicting goals:

trade-off

exploitation ⟷ **exploration**

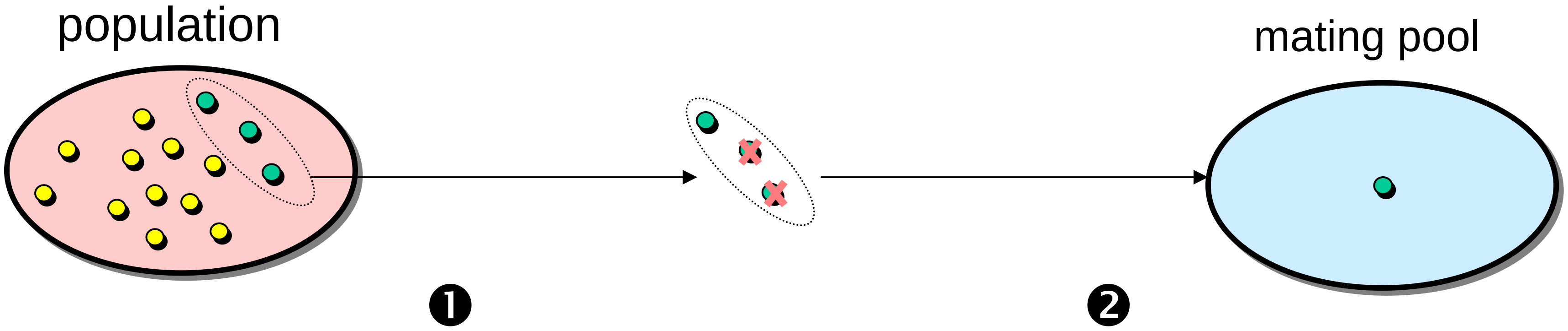
(converge fast) (avoid getting stuck)

Two types of selection:

- **mating selection** = select for variation
- **environmental selection** = select for survival

Example: Tournament Selection

= integrated sampling rate assignment and sampling



➊ uniformly choose
T individuals at
random independently
of fitness

➋ compare fitness
and copy best
individual
in mating pool

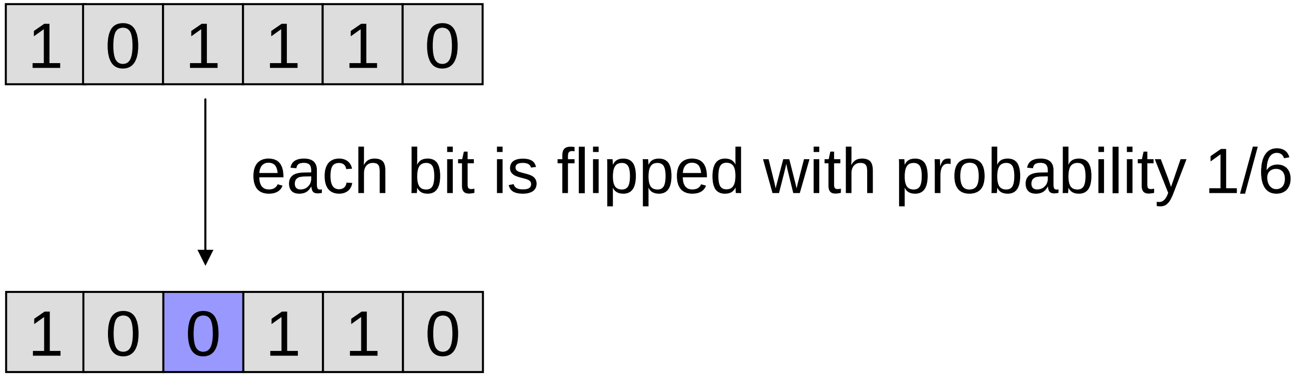
T = tournament size (binary tournament selection means T=2)

Topics

- Multi-Objective Optimization
  - Introduction
  - Multi-Objective Optimization
  - Algorithm Design
    - Design Choices
    - Representation
    - Fitness and Selection
    - ***Variation Operators***
  - Implementation
- Design Space Exploration
  - System Design
  - Problem Specification
  - Example

Example: Vector Mutation

Bit vectors:

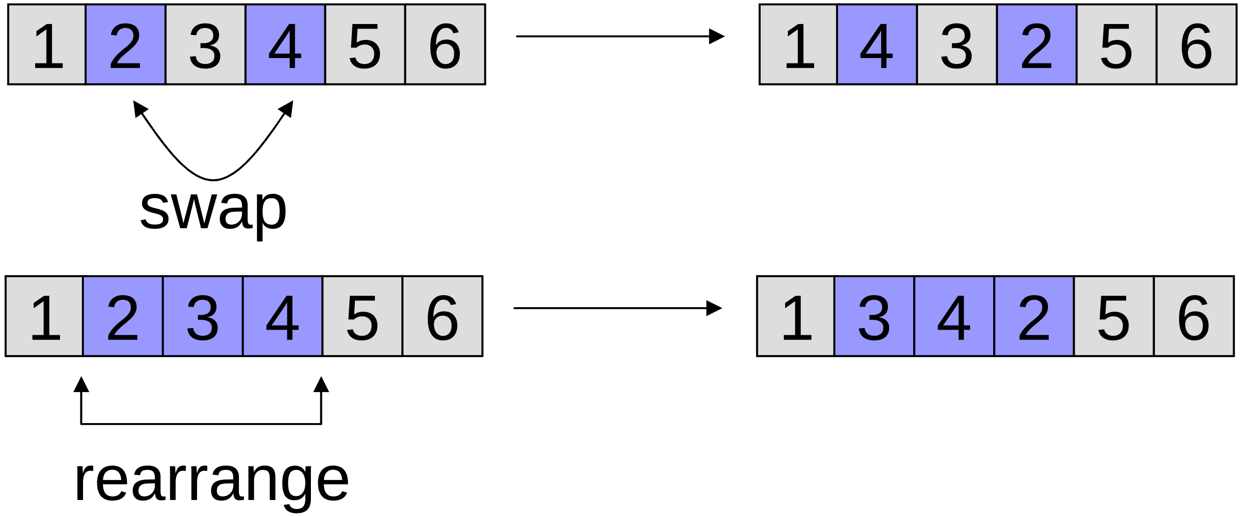


1 0 1 1 1 0

each bit is flipped with probability 1/6

1 0 0 1 1 0

Permutations:



1 2 3 4 5 6 → 1 4 3 2 5 6

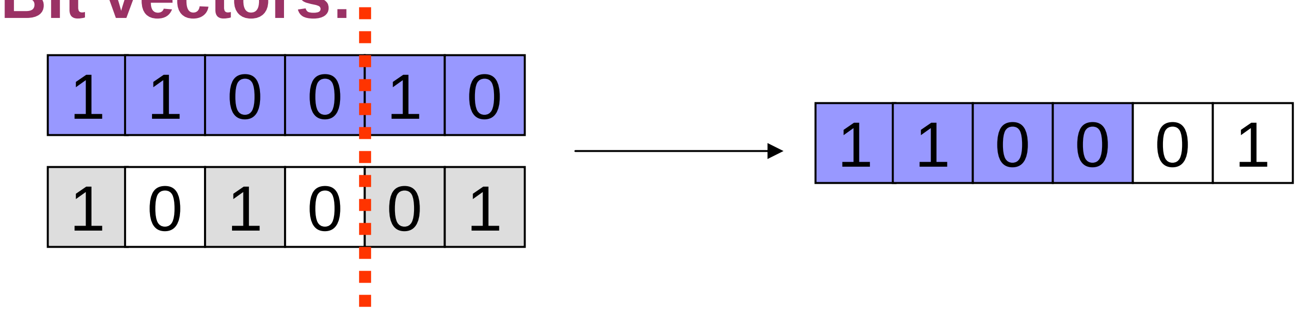
swap

1 2 3 4 5 6 → 1 3 4 2 5 6

rearrange

Example: Vector Recombination

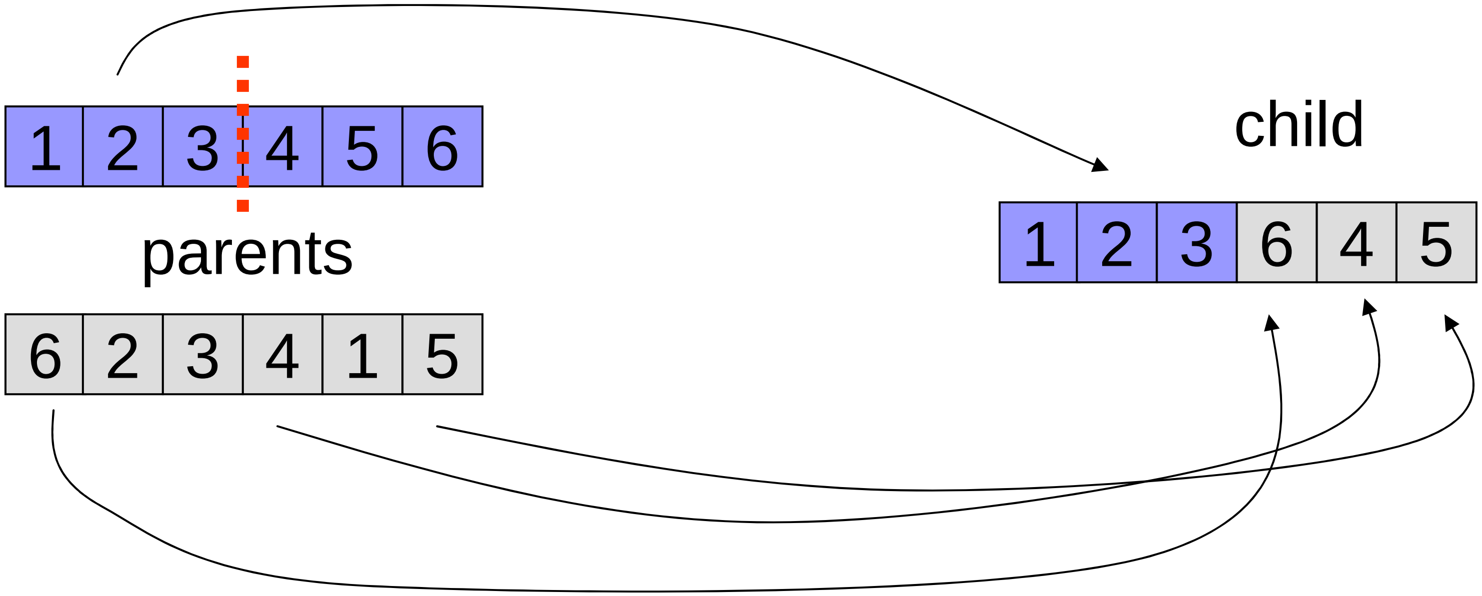
Bit vectors:



1	1	0	0	1	0
1	0	1	0	0	1

1	1	0	0	0	1

Permutations:



1	2	3	4	5	6

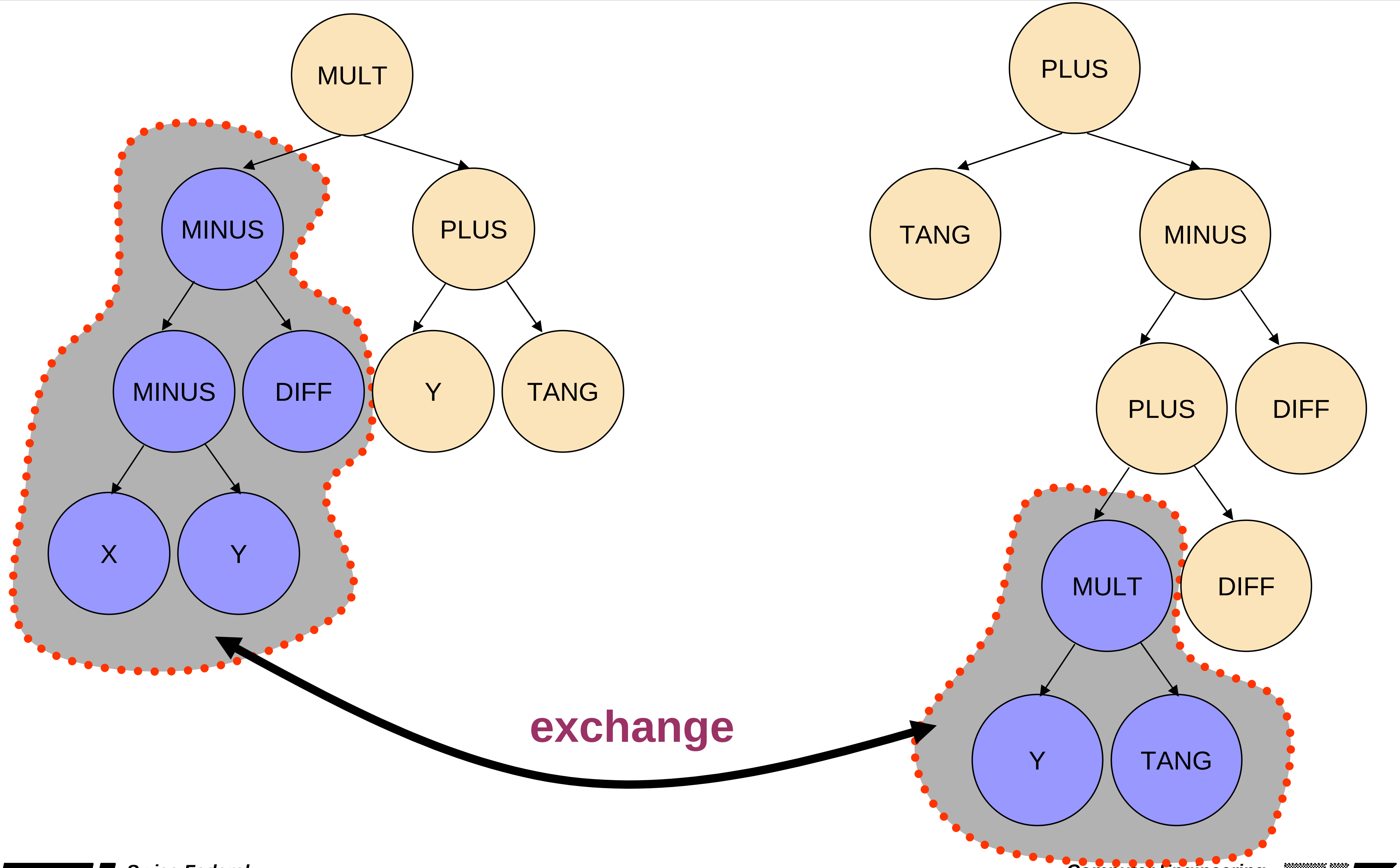
parents

6	2	3	4	1	5

child

1	2	3	6	4	5

Example: Recombination of Trees

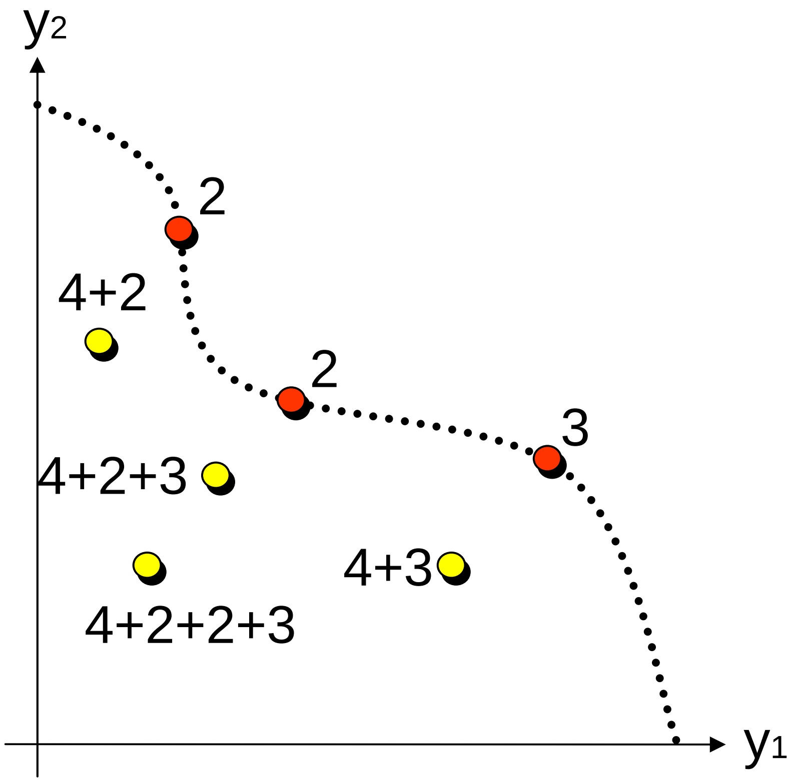


Example: SPEA2 Algorithm

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.

SPEA2 Fitness Assignment

Idea (Step 2): calculate dominance rank weighted by dominance count



non-dominated solutions:

- F = #dominated solutions

dominated solutions

- F = # of non-Pareto solutions + $\sum$ strengths of dominators

Note: lower raw fitness = better quality

Topics

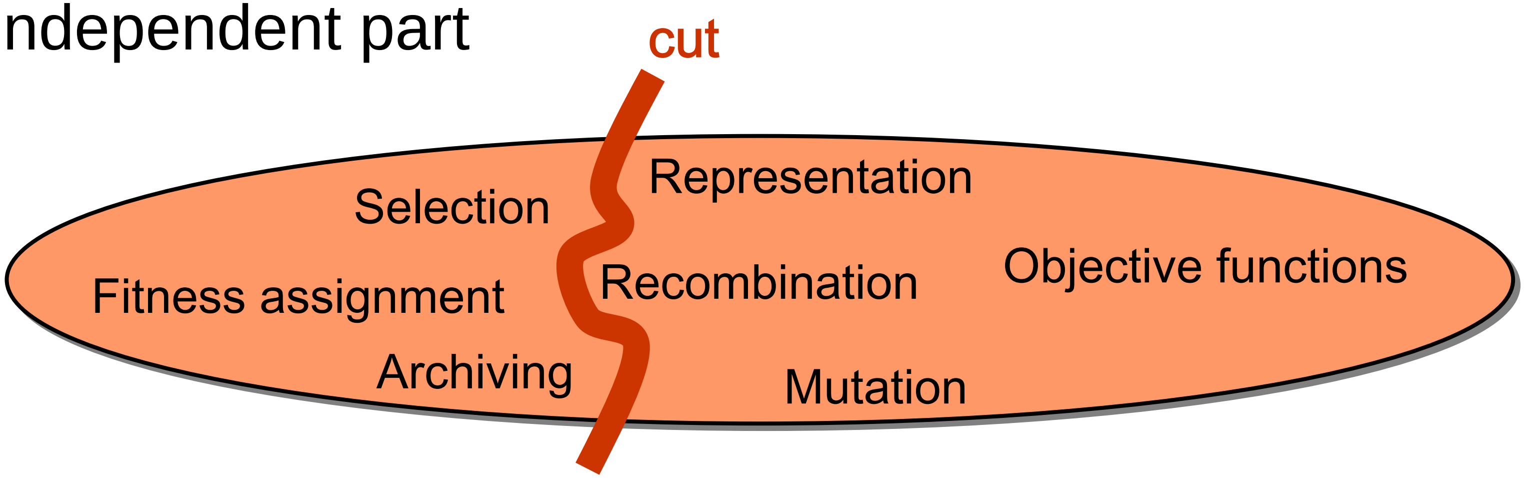
- Multi-Objective Optimization
  - Introduction
  - Multi-Objective Optimization
  - Algorithm Design
    - Design Choices
    - Representation
    - Fitness and Selection
    - Variation Operators
  - ***Implementation***
- Design Space Exploration
  - System Design
  - Problem Specification
  - Example

What Is Needed…

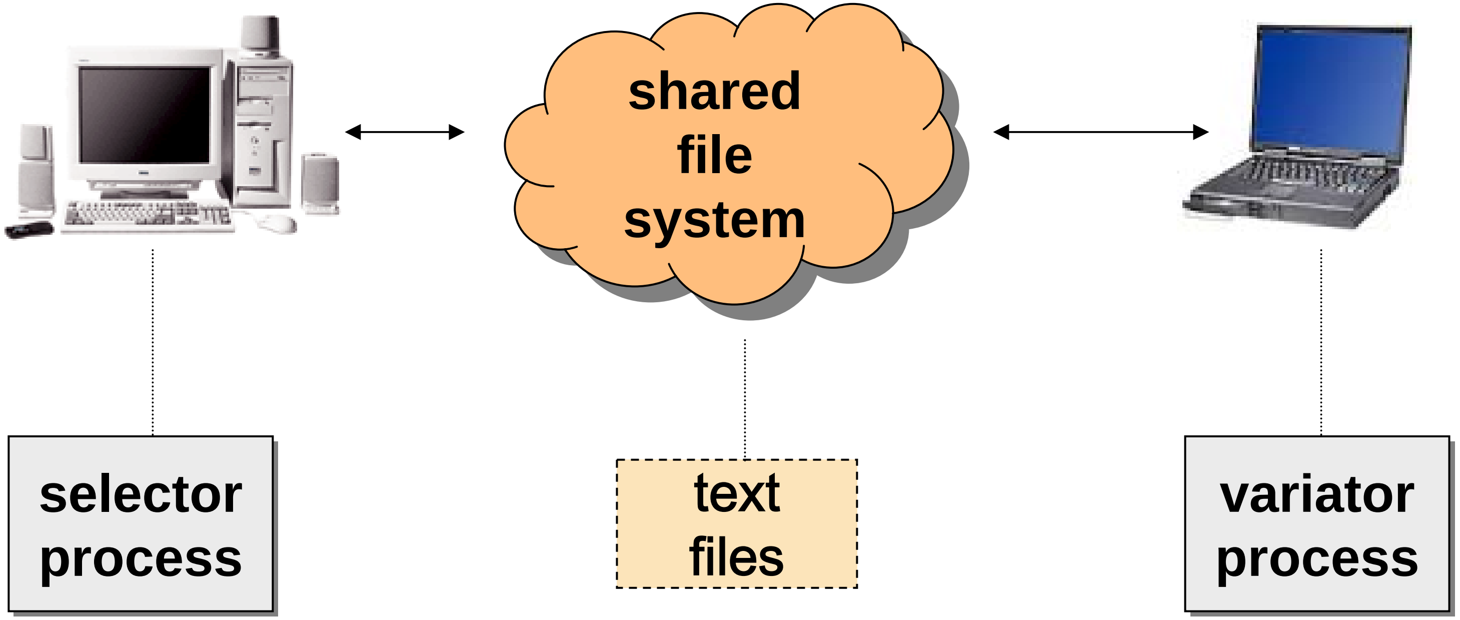
A framework that

- Provides ready-to-use modules (algorithms / applications)
- Is simple to use
- Is independent of programming language and OS
- Comes with minimum overhead

Idea: separate problem-dependent from problem-independent part



PISA: Implementation



shared file system

selector process

text files

variator process

application independent:

- mating / environmental selection
- individuals are described
  by IDs and objective vectors

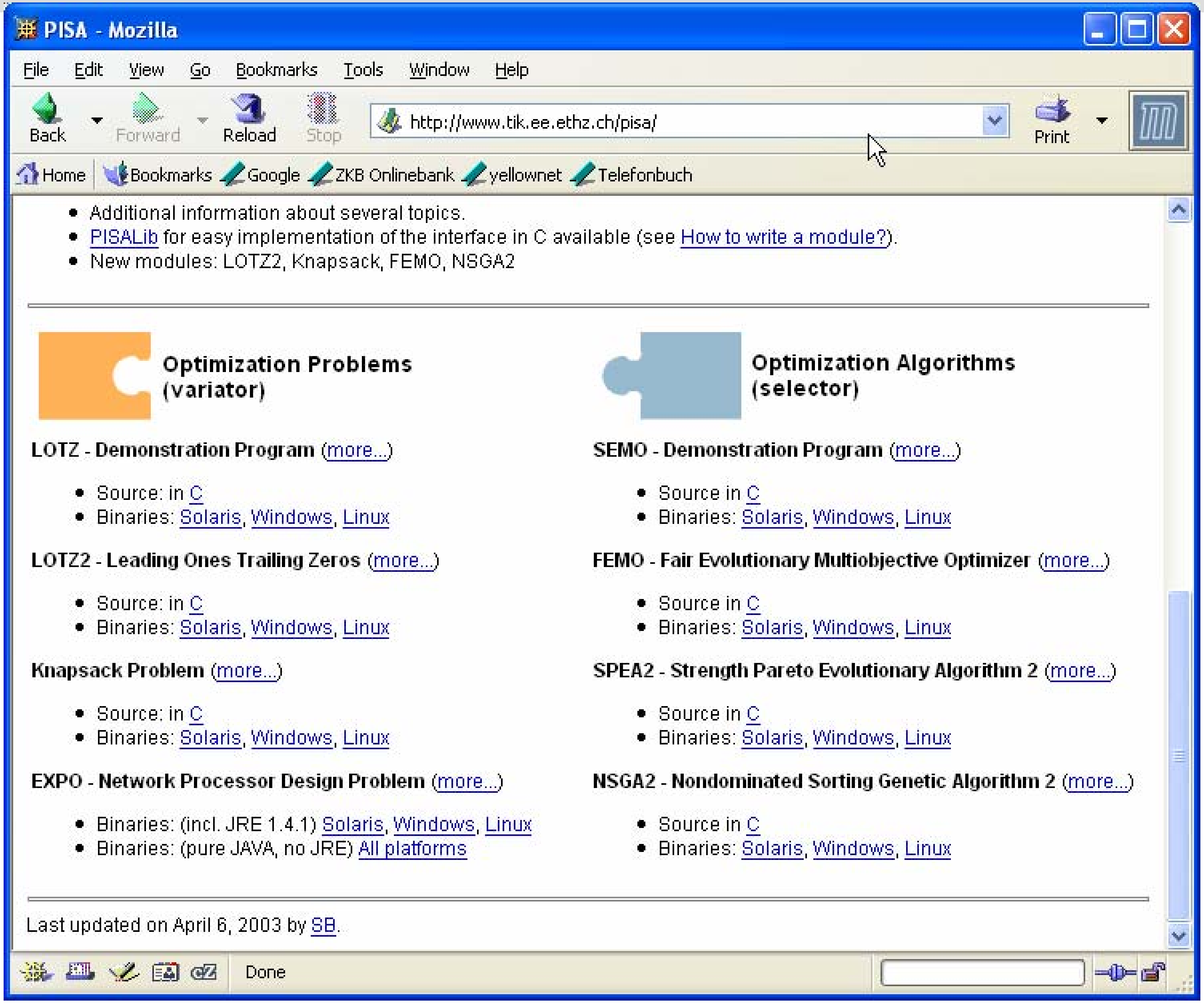
handshake protocol:

- state / action
- individual IDs
- objective vectors
- parameters

application dependent:

- variation operators
- stores and manages individuals

PISA Installation



- Additional information about several topics.
- PISALib for easy implementation of the interface in C available (see How to write a module?).
- New modules: LOTZ2, Knapsack, FEMO, NSGA2

Optimization Problems (variator)

LOTZ - Demonstration Program (more...)

- Source: in C
- Binaries: Solaris, Windows, Linux

LOTZ2 - Leading Ones Trailing Zeros (more...)

- Source: in C
- Binaries: Solaris, Windows, Linux

Knapsack Problem (more...)

- Source: in C
- Binaries: Solaris, Windows, Linux

EXPO - Network Processor Design Problem (more...)

- Binaries: (incl. JRE 1.4.1) Solaris, Windows, Linux
- Binaries: (pure JAVA, no JRE) All platforms

Optimization Algorithms (selector)

SEMO - Demonstration Program (more...)

- Source in C
- Binaries: Solaris, Windows, Linux

FEMO - Fair Evolutionary Multiobjective Optimizer (more...)

- Source in C
- Binaries: Solaris, Windows, Linux

SPEA2 - Strength Pareto Evolutionary Algorithm 2 (more...)

- Source in C
- Binaries: Solaris, Windows, Linux

NSGA2 - Nondominated Sorting Genetic Algorithm 2 (more...)

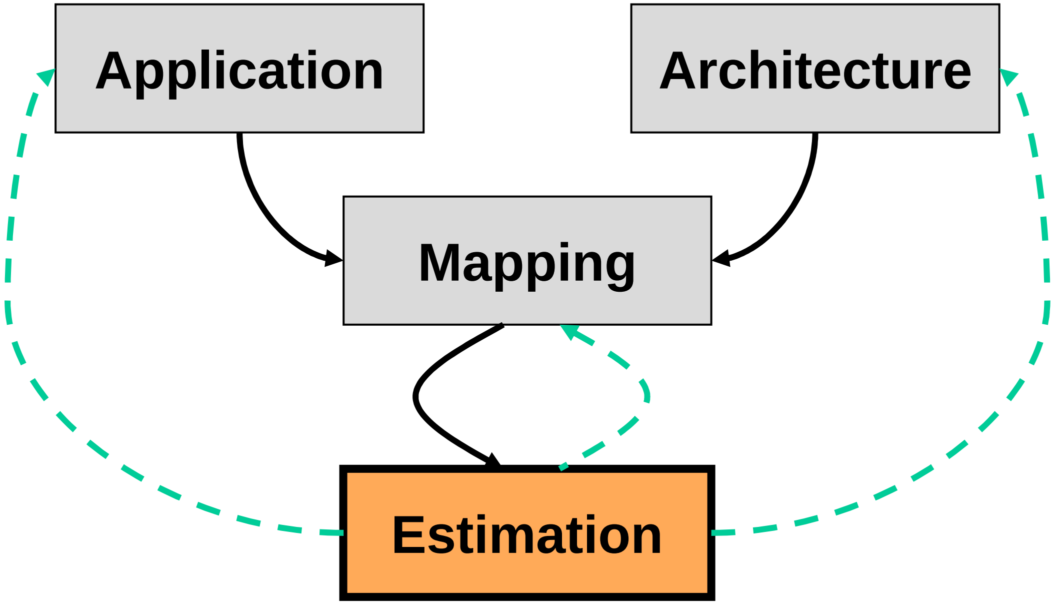
- Source in C
- Binaries: Solaris, Windows, Linux

Last updated on April 6, 2003 by SB.

Topics

- Multi-Objective Optimization
  - Introduction
  - Multi-Objective Optimization
  - Algorithm Design
    - Design Choices
    - Representation
    - Fitness and Selection
    - Variation Operators
  - Implementation
- ***Design Space Exploration***
  - ***System Design***
  - Problem Specification
  - Example

Design Space Exploration



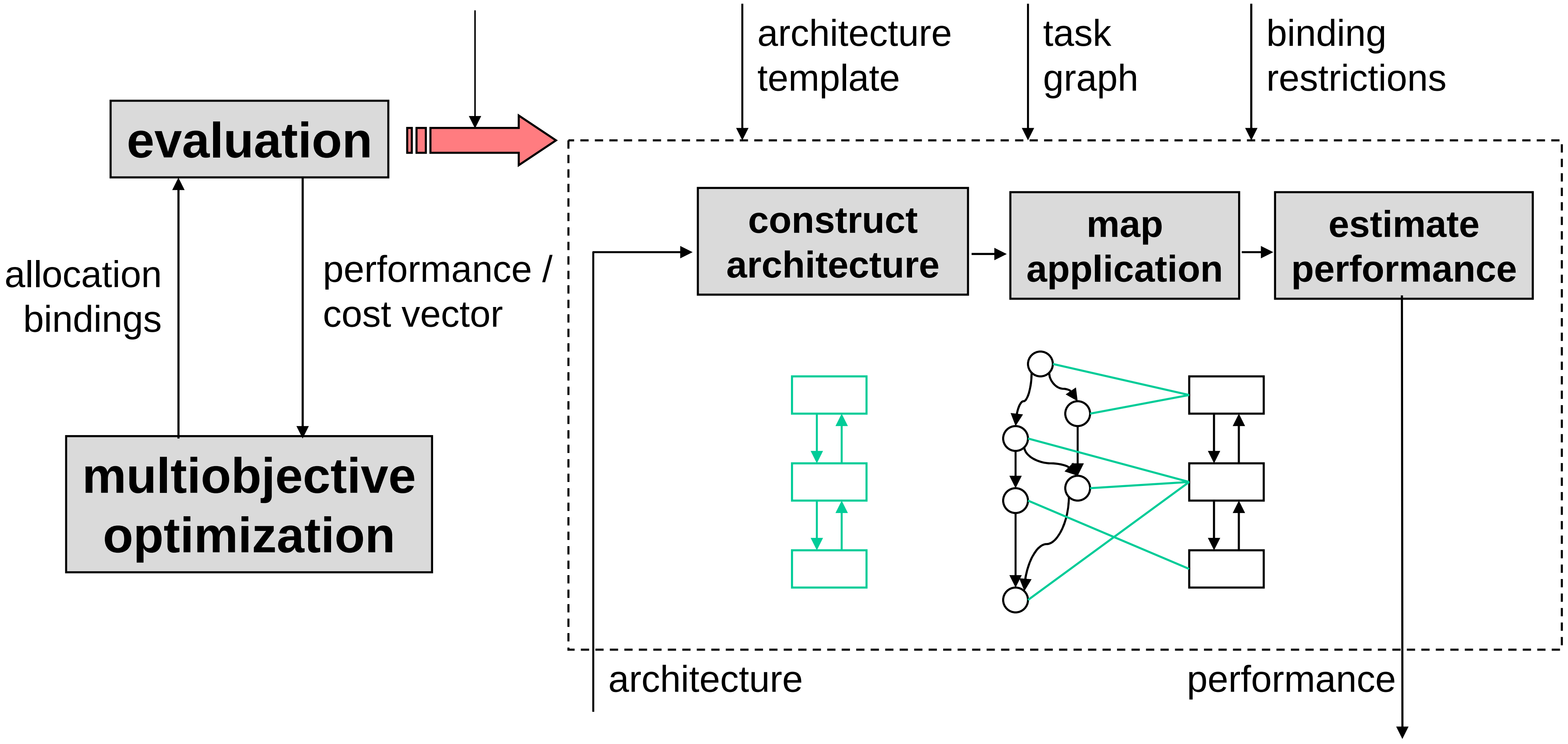
Application

Architecture

Mapping

Estimation

Embedded System Design



architecture template

task graph

binding restrictions

evaluation

allocation bindings

performance / cost vector

multiobjective optimization

construct architecture

map application

estimate performance

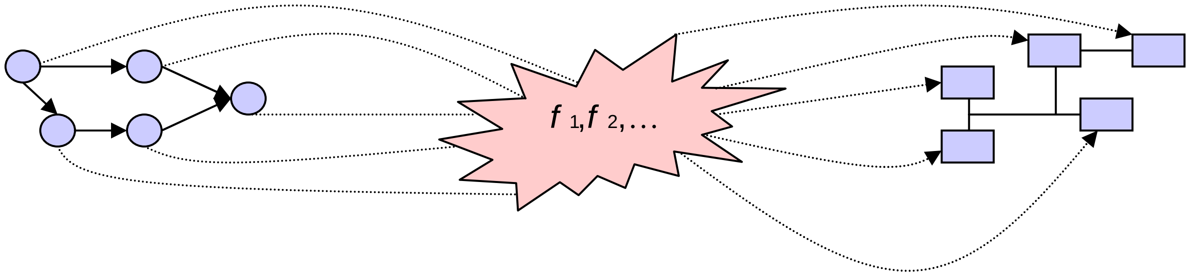
architecture

performance

Example: System Synthesis

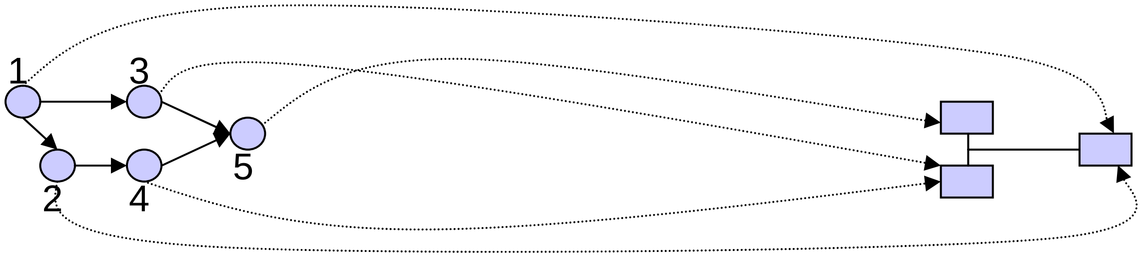
Given:

algorithm mappings architectures



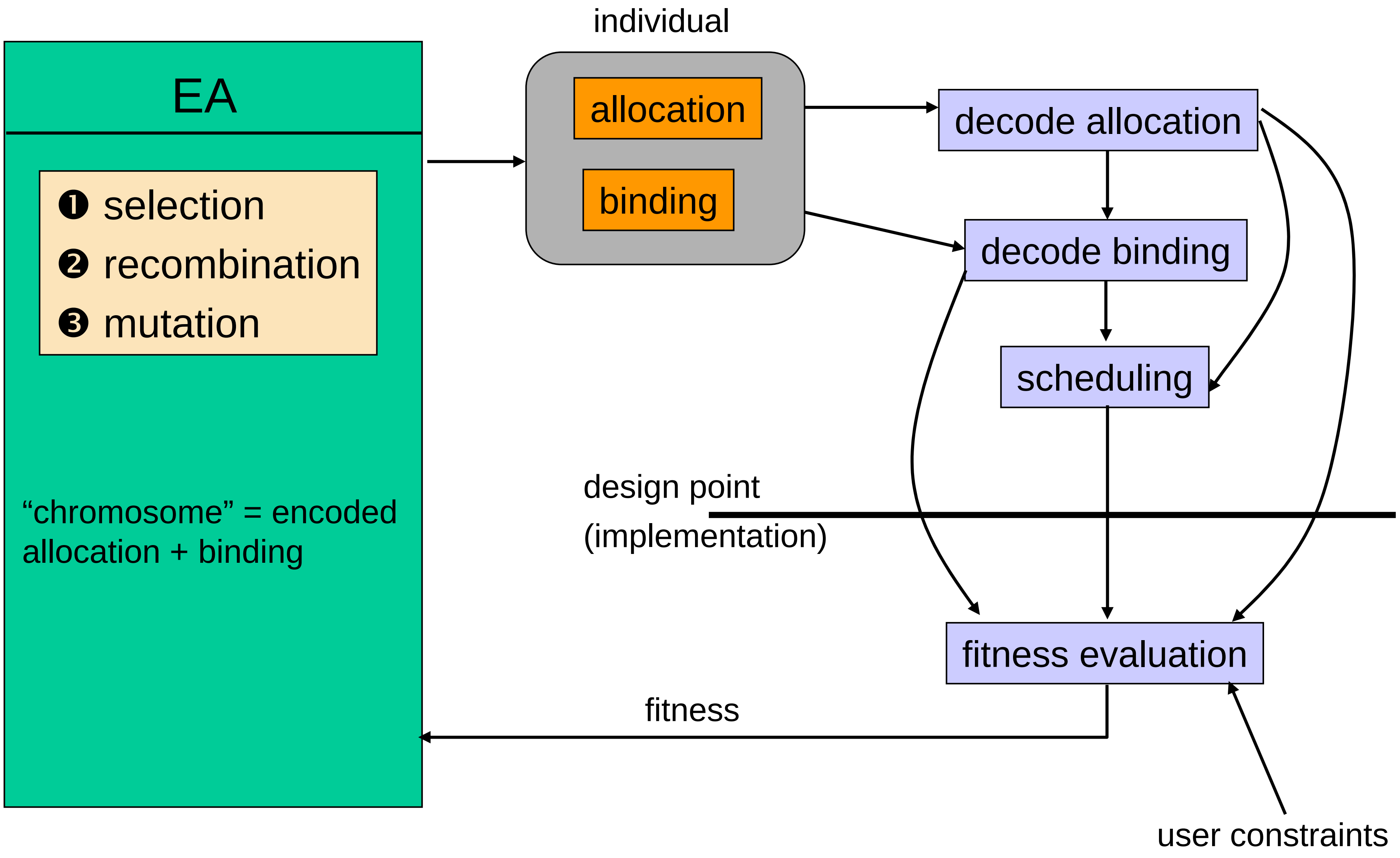
Goal:

schedule mapping architecture



Objectives: cost, latency, power consumption

Evolutionary Algorithms for DSE

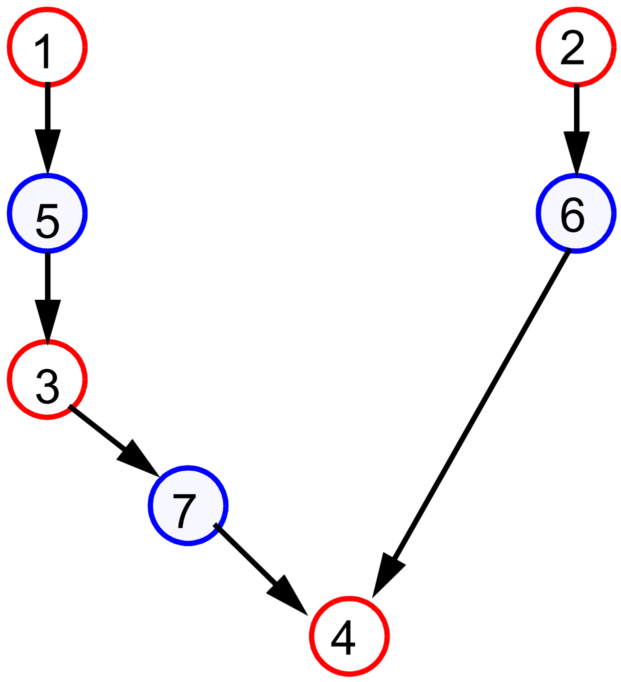


Topics

- Multi-Objective Optimization
  - Introduction
  - Multi-Objective Optimization
  - Algorithm Design
    - Design Choices
    - Representation
    - Fitness and Selection
    - Variation Operators
  - Implementation
- Design Space Exploration
  - System Design
  - ***Problem Specification***
  - Example

Basic Model – Problem Graph

Problem graph $G_P(V_P,E_P)$:

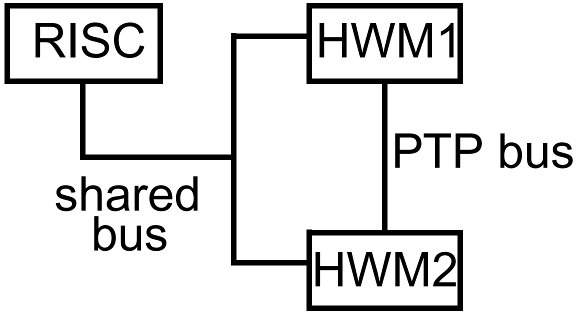


Interpretation:

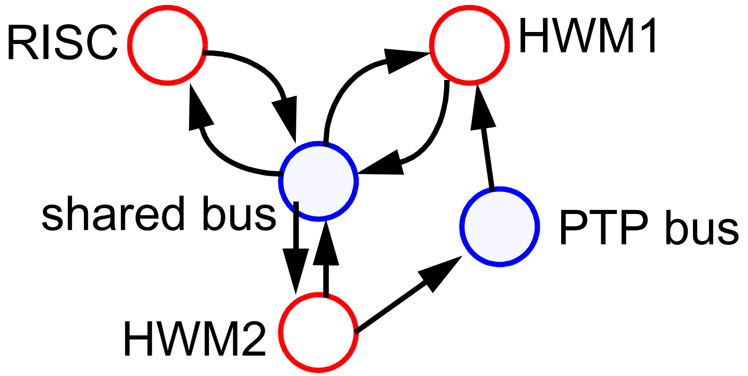
- V_P consists of functional nodes V_P^f (task, procedure) and communication nodes V_P^c .
- E_P represent data dependencies

Basic model – architecture graph

Architecture graph $G_A(V_A,E_A)$:



Architecture

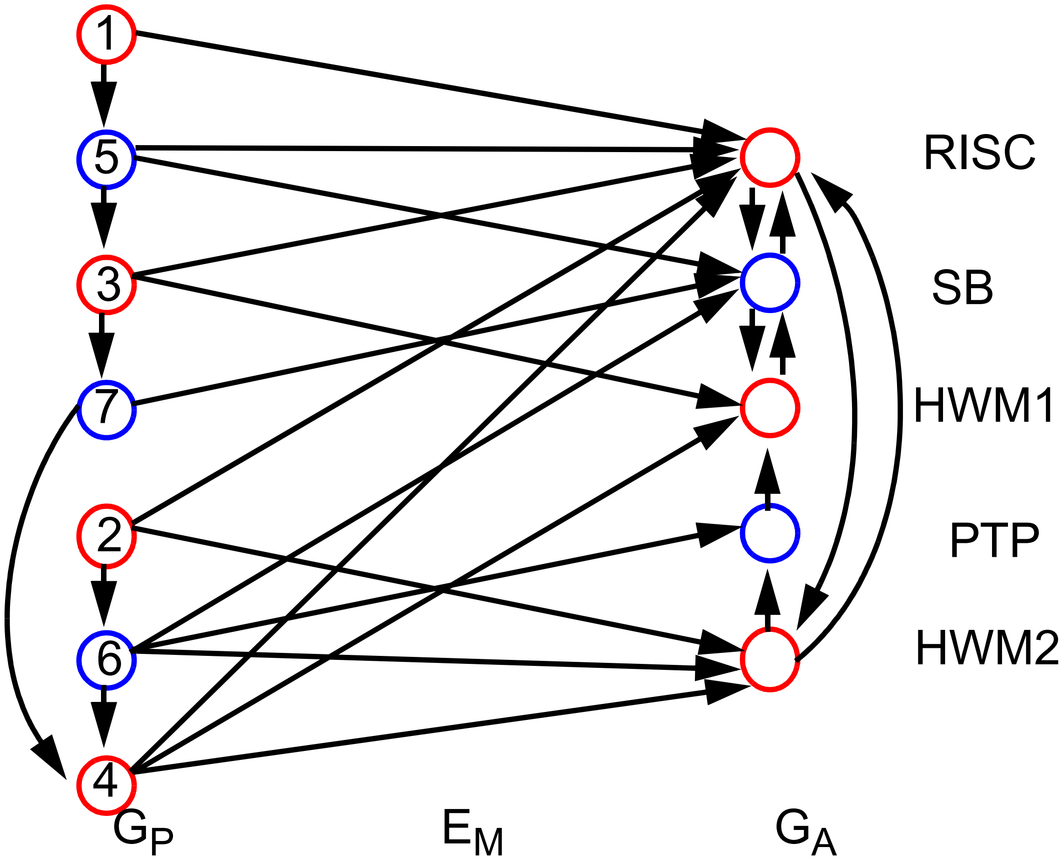


Architecture graph

- V_A consists of functional resources V_A^f (RISC, ASIC) and bus resources V_A^c. These components are potentially allocatable.
- E_A model directed communication.

Basic model – specification graph

Definition: A specification graph is a graph $G_S=(V_S,E_S)$ consisting of a problem graph G_P, an architecture graph G_A, and edges E_M. In particular, $V_S=V_P \cup V_A$, $E_S=E_P \cup E_A \cup E_M$



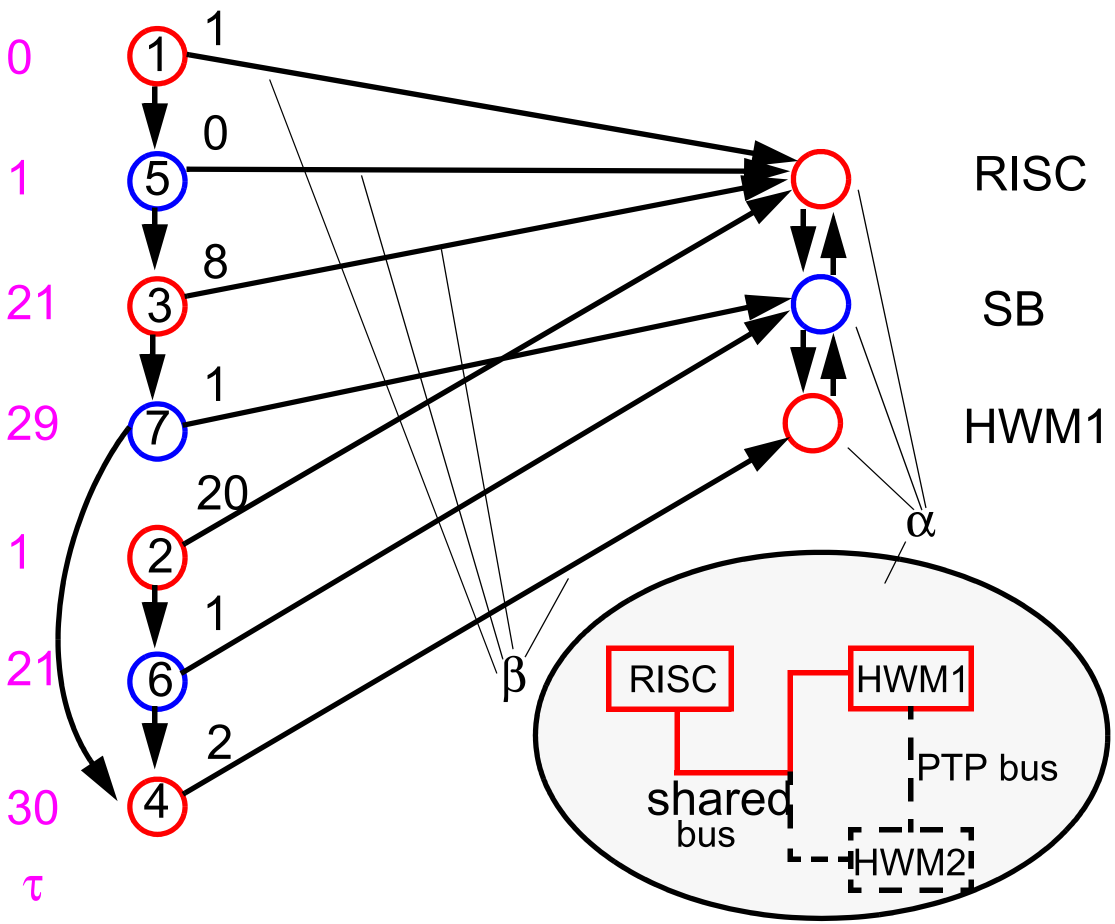
Basic model - synthesis

Three main tasks of synthesis:

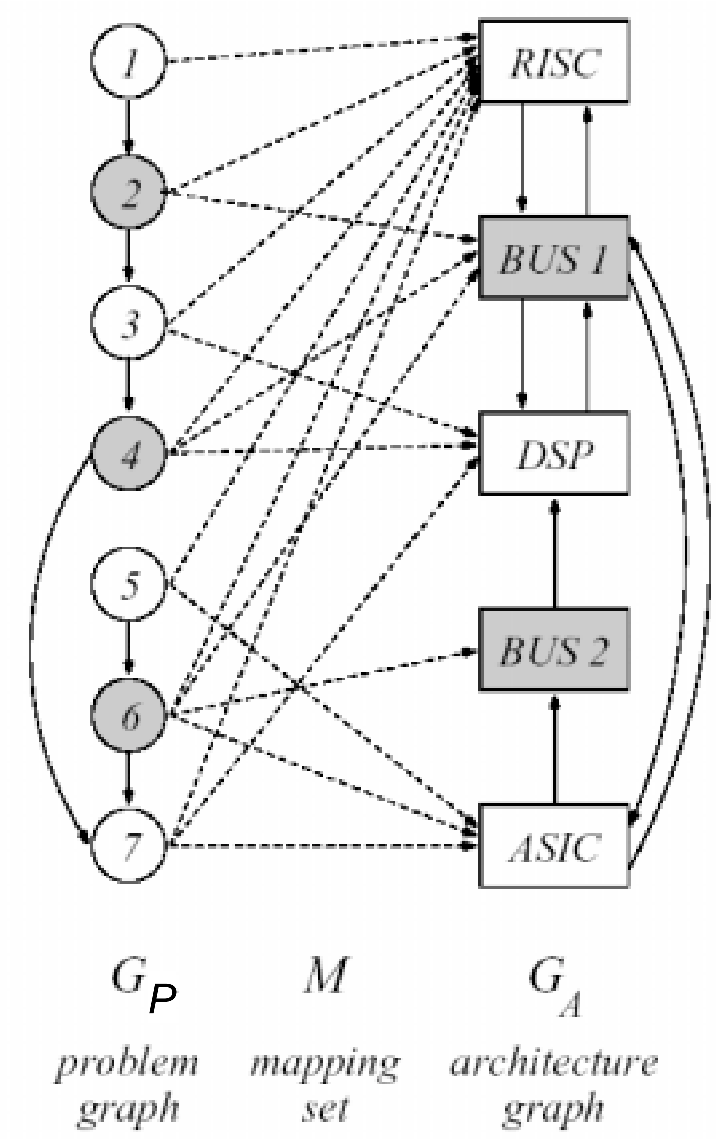
- Allocation α is a subset of V_A.
- Binding β is a subset of E_M, i.e., a mapping of functional nodes of V_P onto resource nodes of V_A.
- Schedule τ is a function that assigns a number (start time) to each functional node.

Basic model - implementation

Definition: Given a specification graph G_S an implementation is a triple (α,β,τ), where α is a feasible allocation, β is a feasible binding, and τ is a schedule.



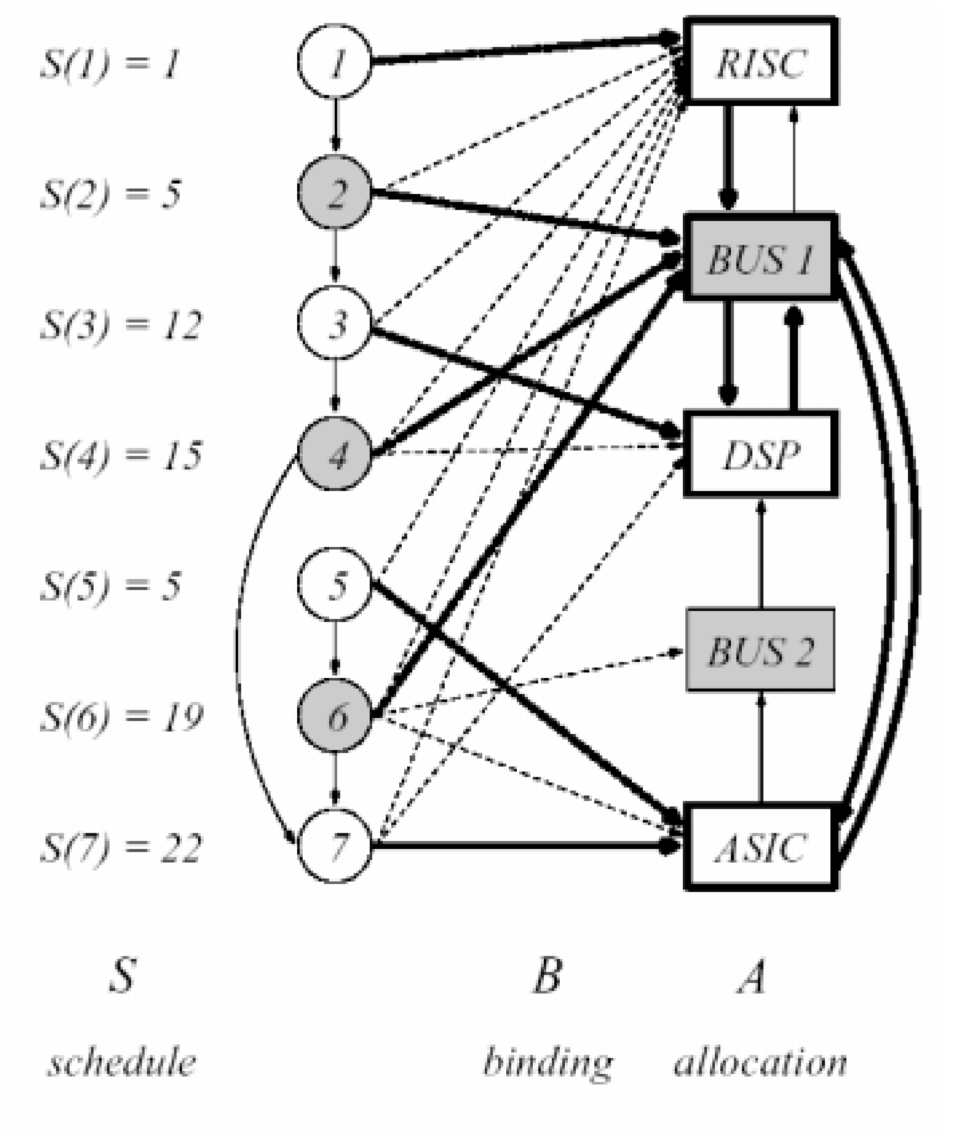
Example



G_P problem graph

M mapping set

G_A architecture graph



$S(1) = 1$

$S(2) = 5$

$S(3) = 12$

$S(4) = 15$

$S(5) = 5$

$S(6) = 19$

$S(7) = 22$

S schedule

B binding

A allocation

Challenges

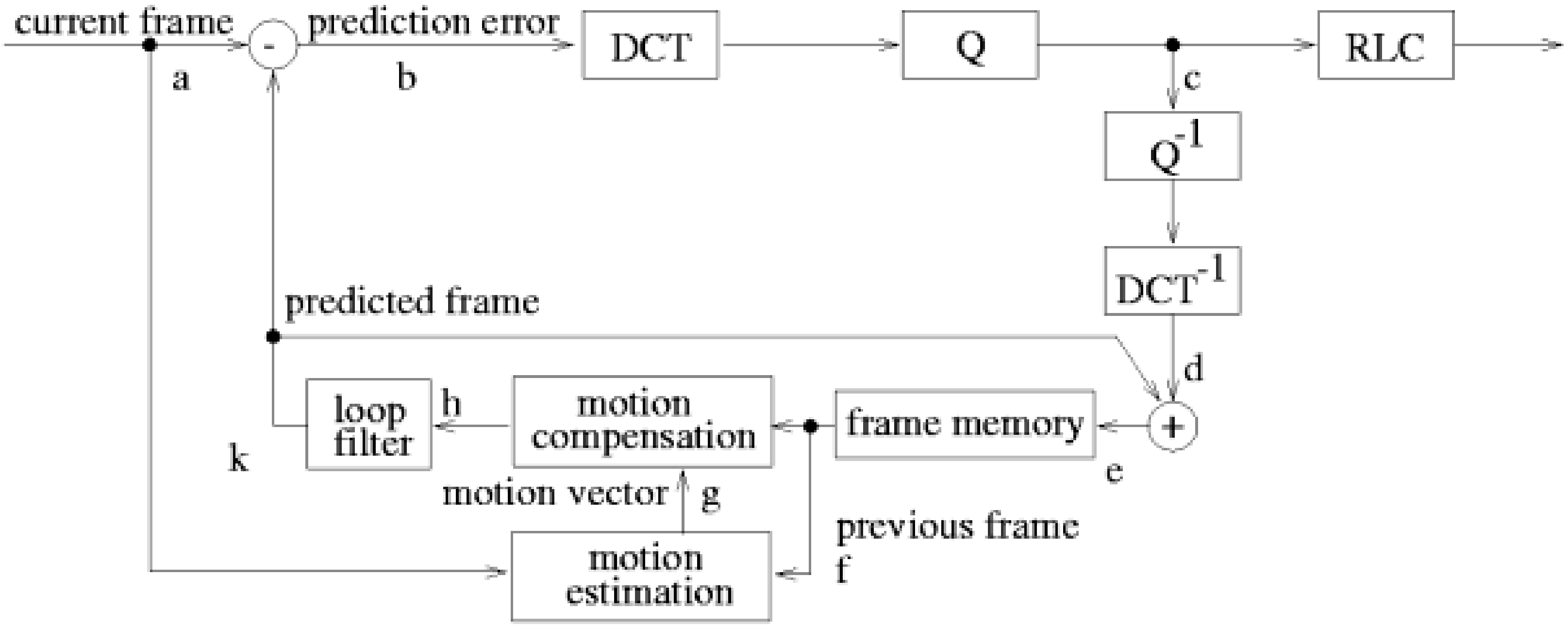
- Encoding of (allocation+binding)
  - simple encoding
    - eg. one bit per resource, one variable per binding
    - easy to implement
    - many infeasible partitionings
  - encoding + <u>repair</u>
    - eg. simple encoding and modify such that for each $v_p \in V_P$ there exists at least one $v_a \in V_A$ with a $\beta(v_p) = v_a$
    - reduces number of infeasible partitionings
- Generation of the initial population, mutation
- Recombination

Topics

- Multi-Objective Optimization
  - Introduction
  - Multi-Objective Optimization
  - Algorithm Design
    - Design Choices
    - Representation
    - Fitness and Selection
    - Variation Operators
  - Implementation
- Design Space Exploration
  - System Design
  - Problem Specification
  - ***Example***

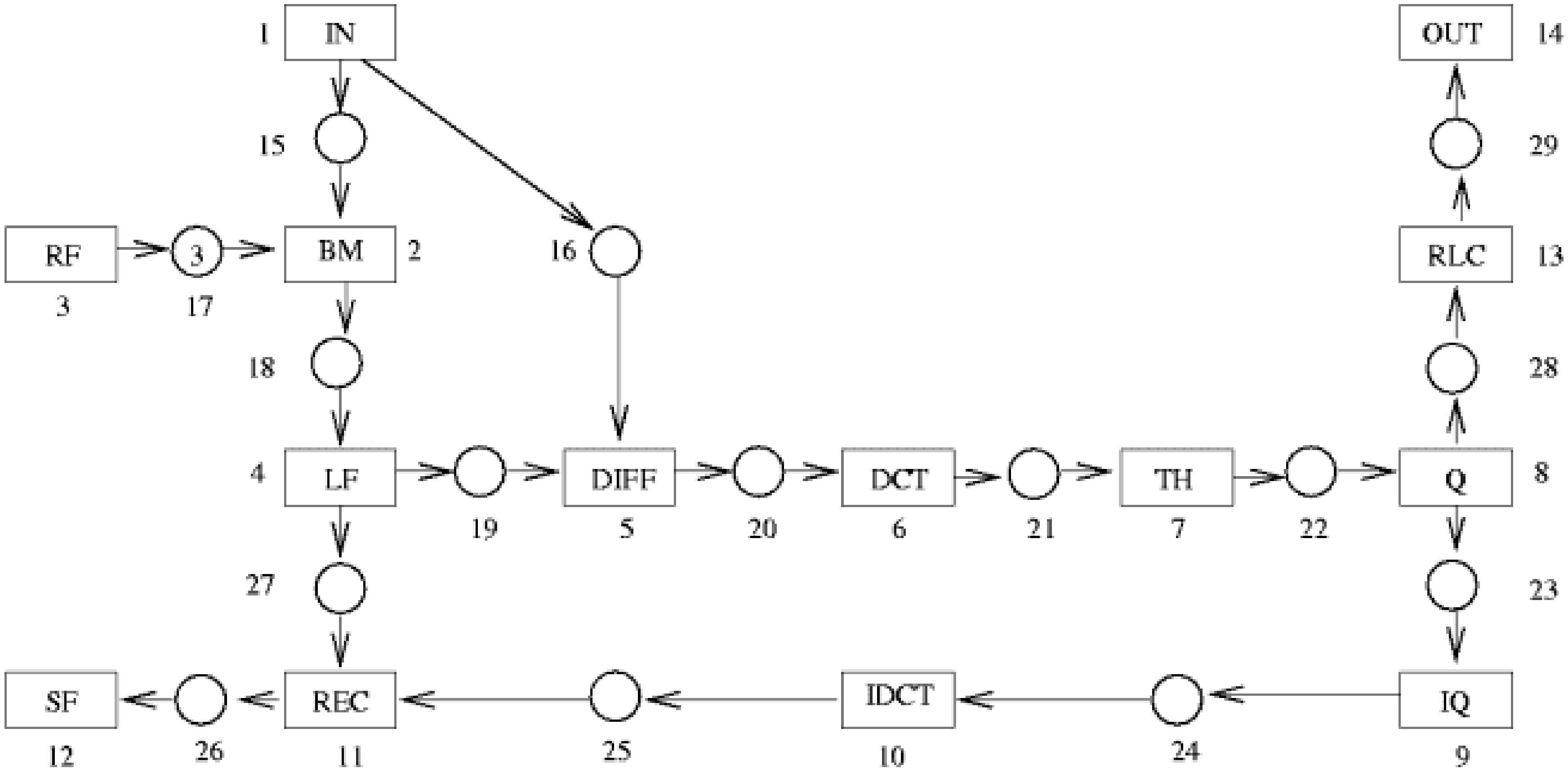
Exploration - Case Study (1)

behavioral specification of a video codec for video compression

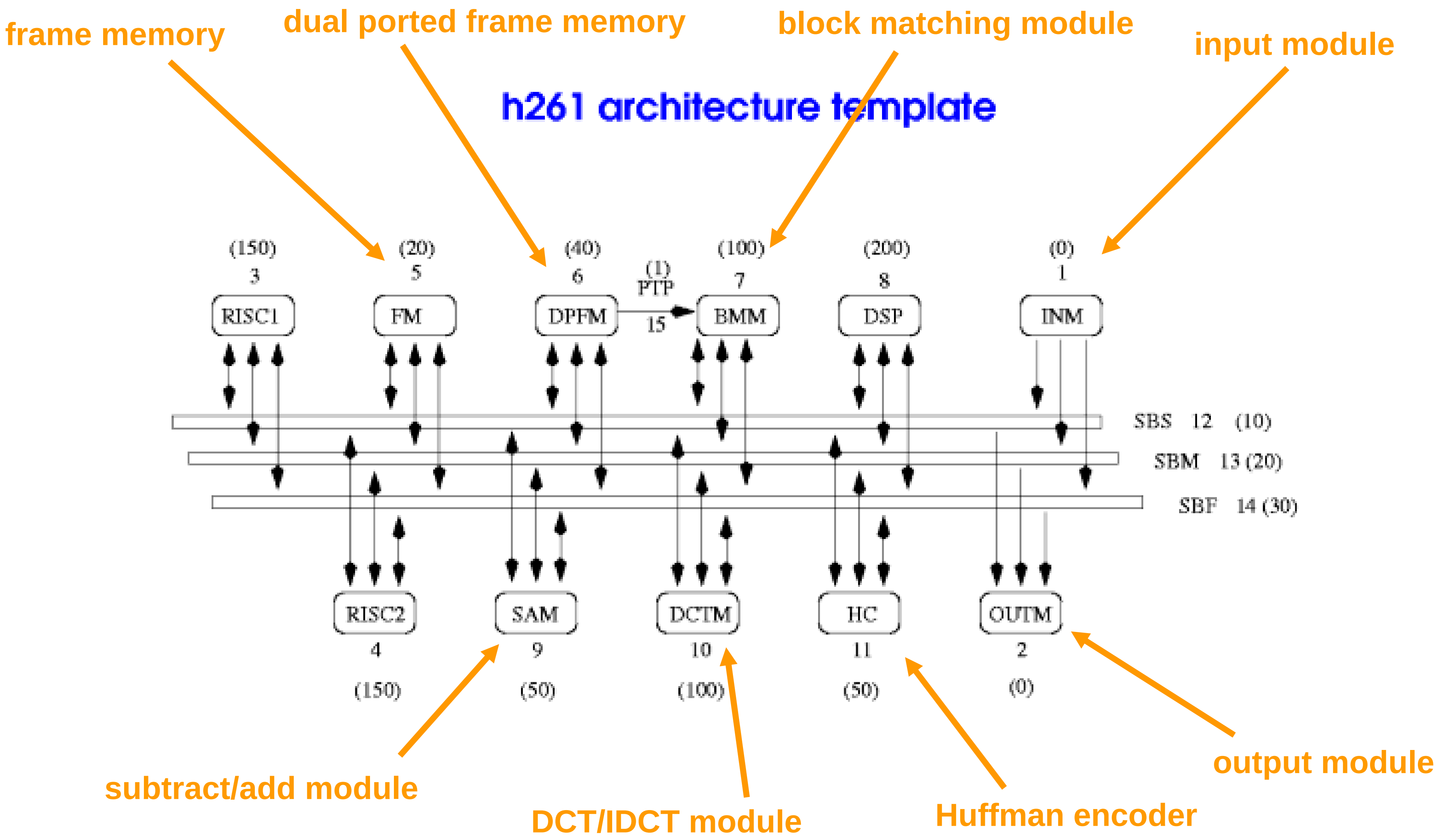


Exploration - Case Study (2)

problem graph of the video coder



Exploration - Case Study (3)



h261 architecture template

frame memory

dual ported frame memory

block matching module

input module

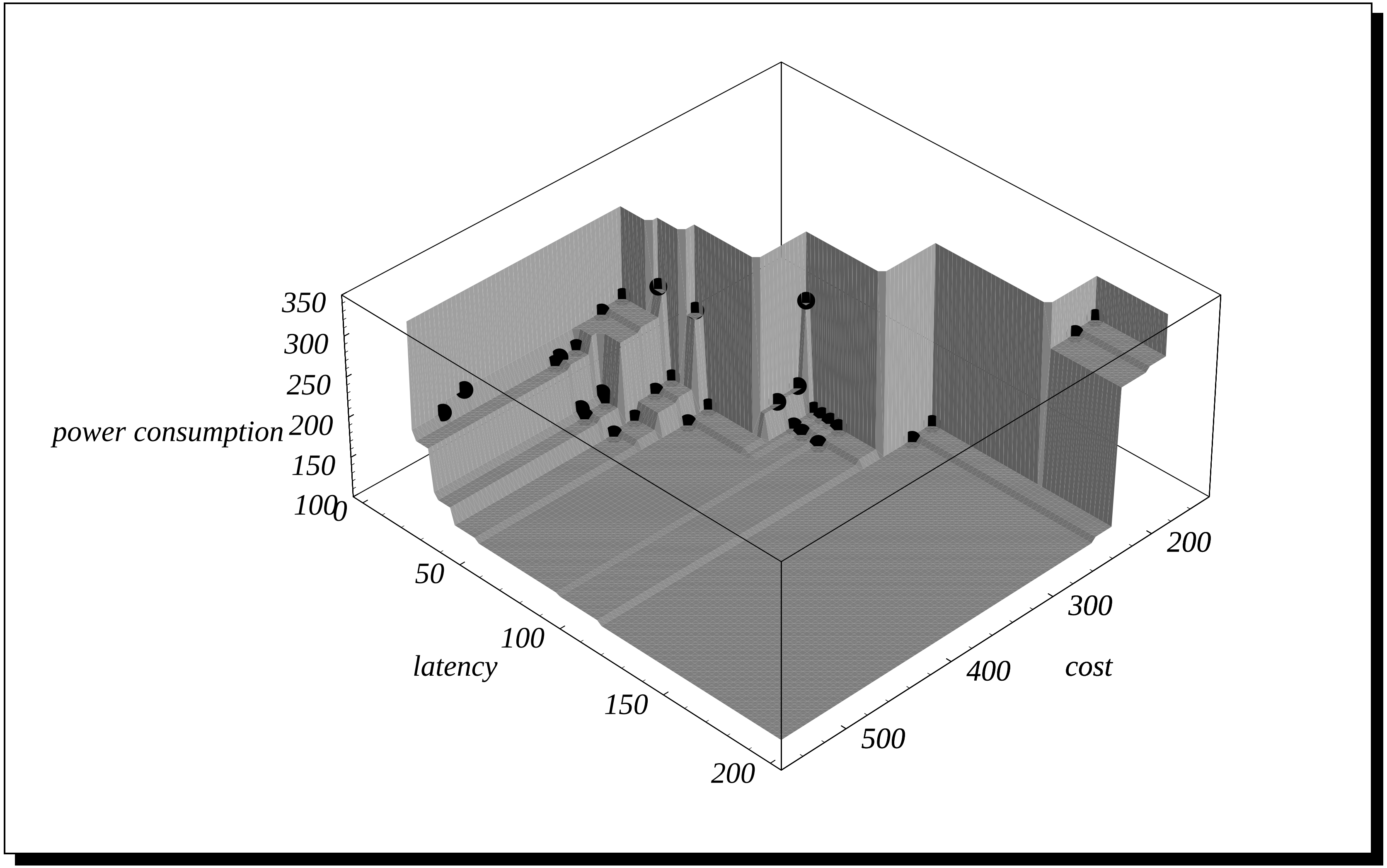
subtract/add module

DCT/IDCT module

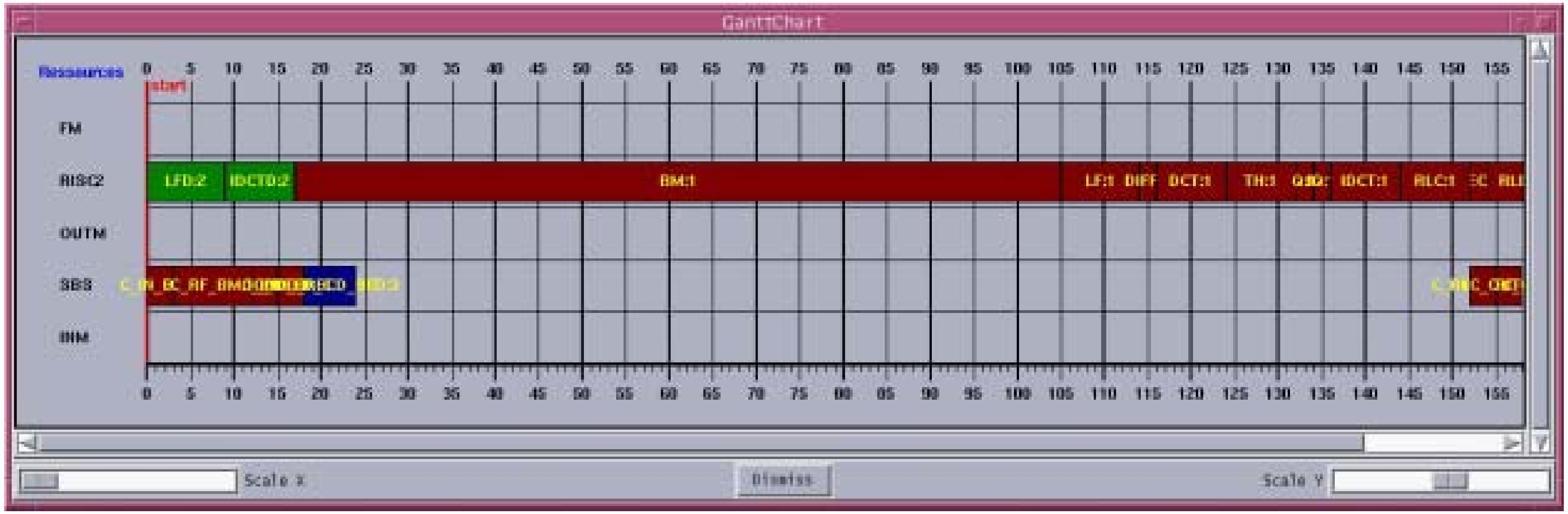
Huffman encoder

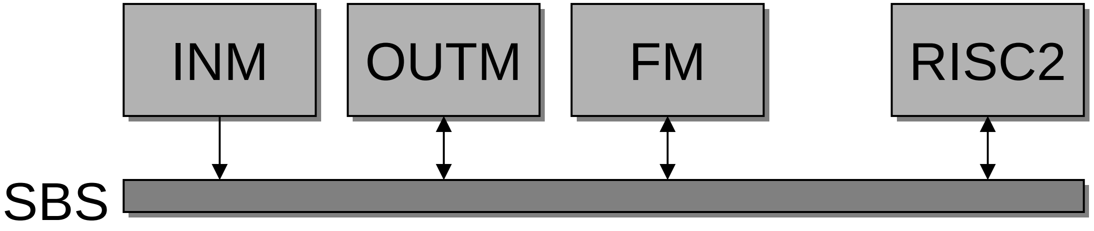
output module

EA Case Study - Design Space



Exploration Case Study - Solution 1





INM

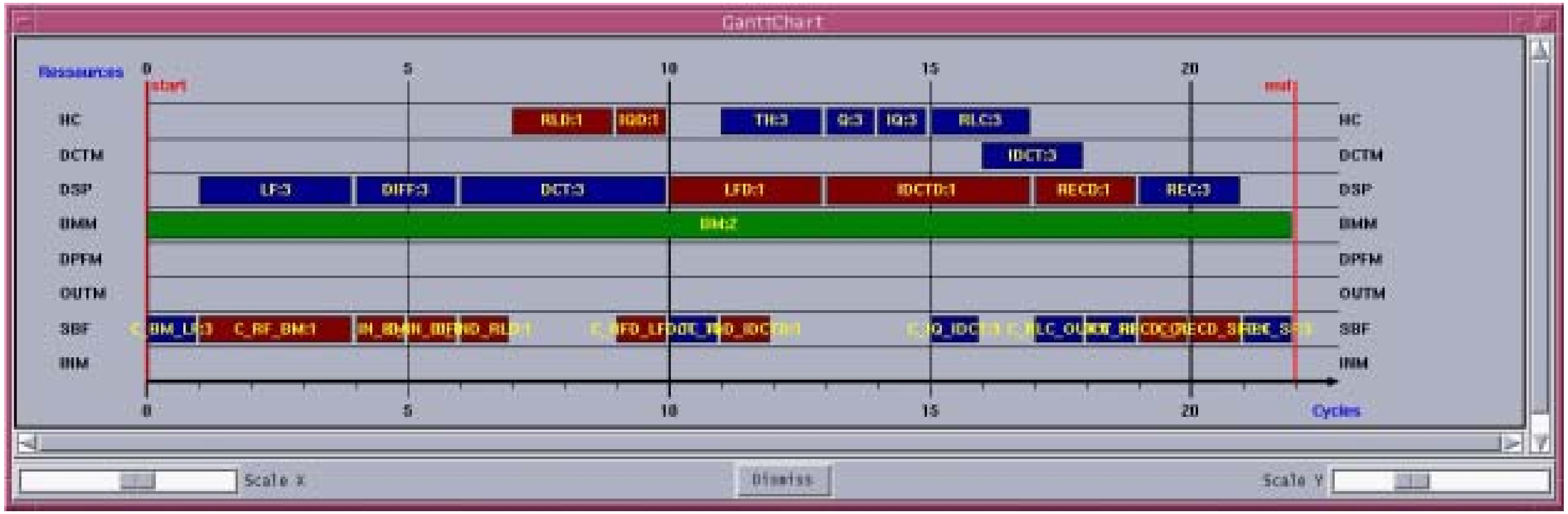
OUTM

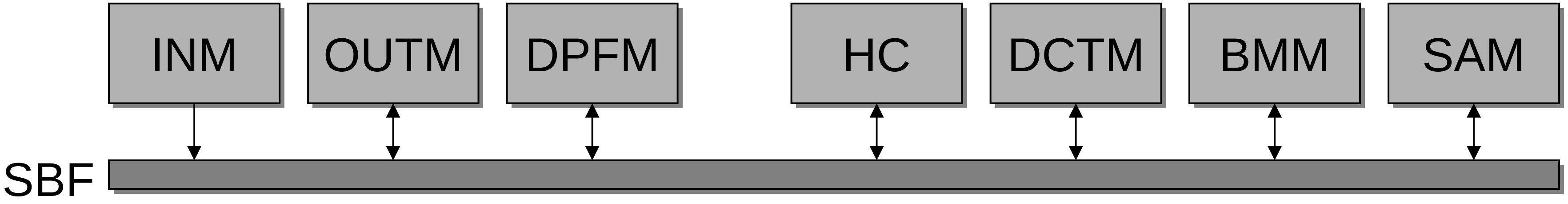
FM

RISC2

SBS

Exploration Case Study - Solution 2




INM

OUTM

DPFM

HC

DCTM

BMM

SAM

SBF