technische universität
dortmund

fakultät für informatik
informatik 12

Synthese Eingebetteter Systeme

Wintersemester 2012/13

16 – Abbildung von Anwendungen: Optimierung mit DOL

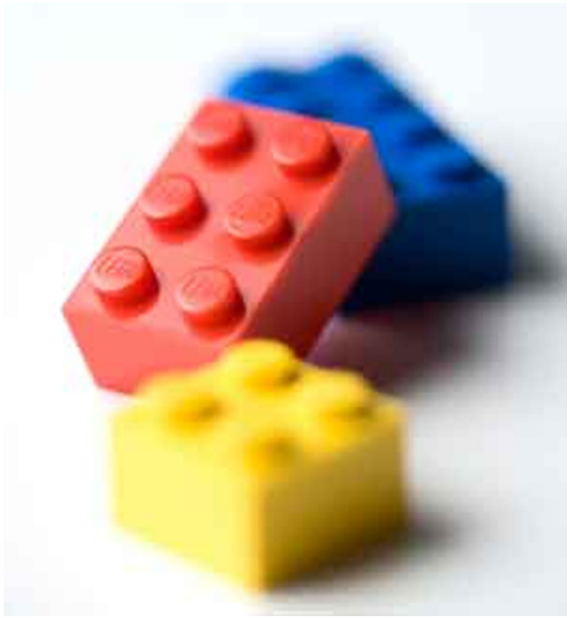
Michael Engel
Informatik 12
TU Dortmund

2013/01/09

– unter Verwendung von Foliensätzen von Prof. Lothar Thiele und Dr. Iuliana Bacivarov, ETH Zürich –

Optimierung mit DOL

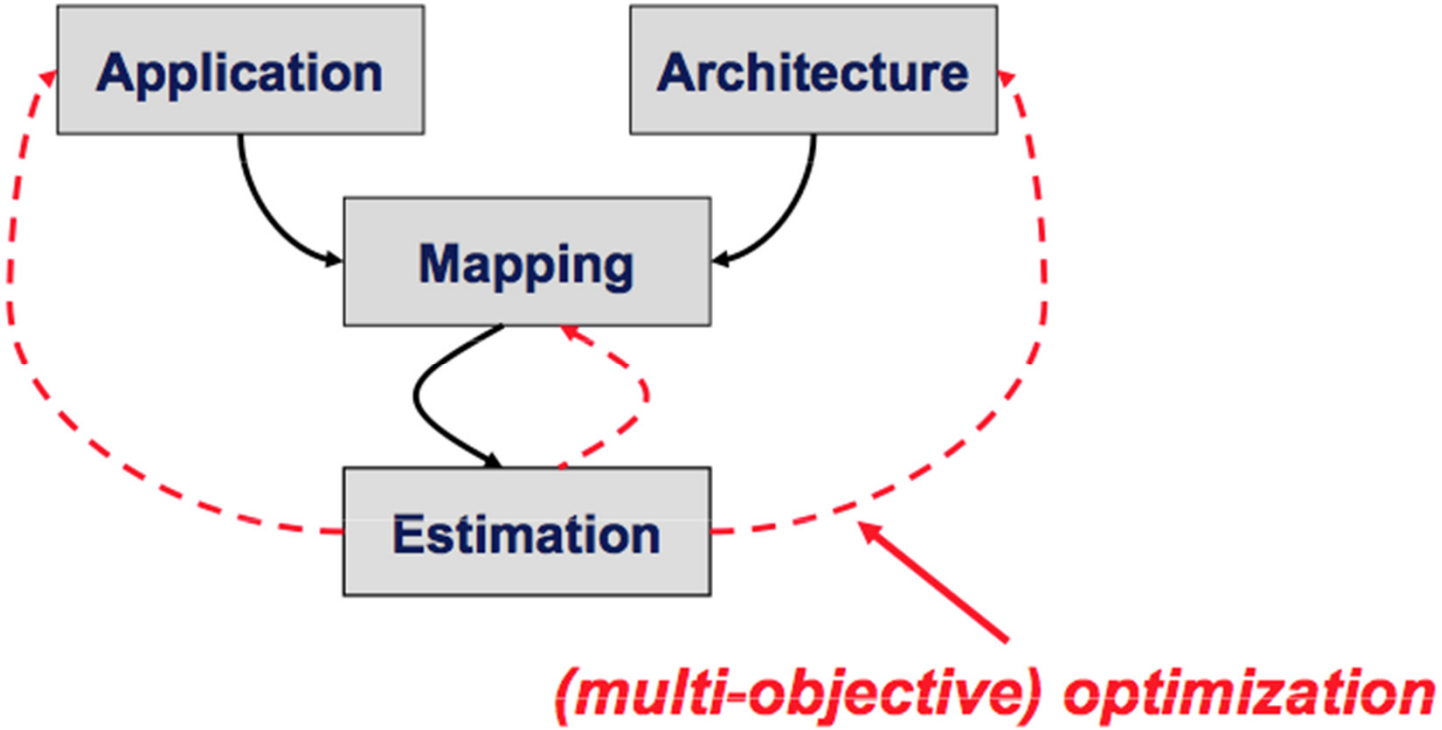
- Multikriterielle Optimierung
  - Pareto-Mengen
  - Evolutionäre Algorithmen
- Optimierung in DOL: SPEA2



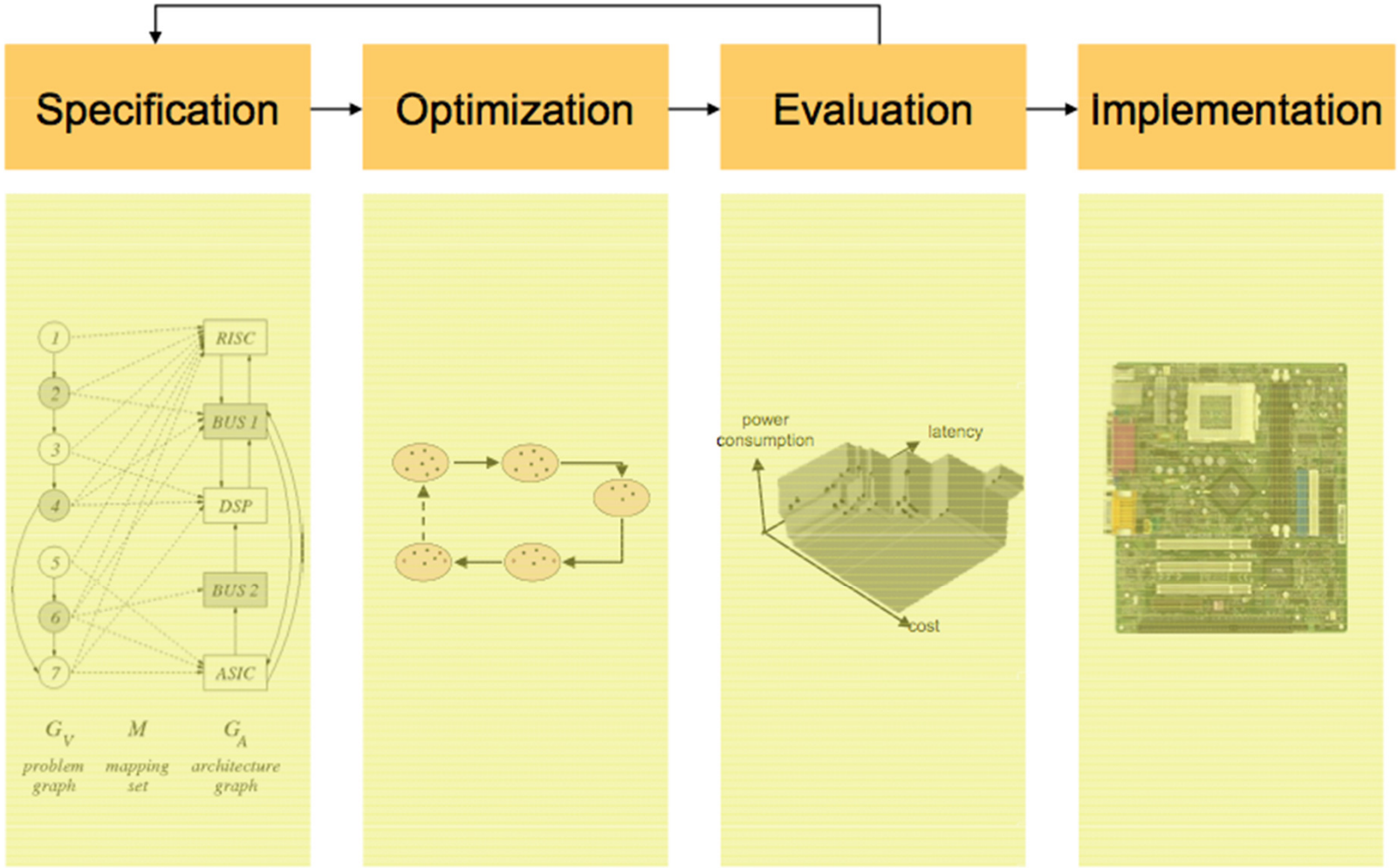
Entwurfsraumerkundung

- Ziel
  - Finde die Pareto-optimalen Abbildungen einer Anwendung auf eine gegebene Architektur
- Probleme
  - Vollständiges Durchsuchen des Entwurfsraums nicht realisierbar
  - Instruktionsgenaue Simulation zu langsam für Entwurfsraumerkundung

Entwurfsraumerkundung



Entwurfsraumerkundung



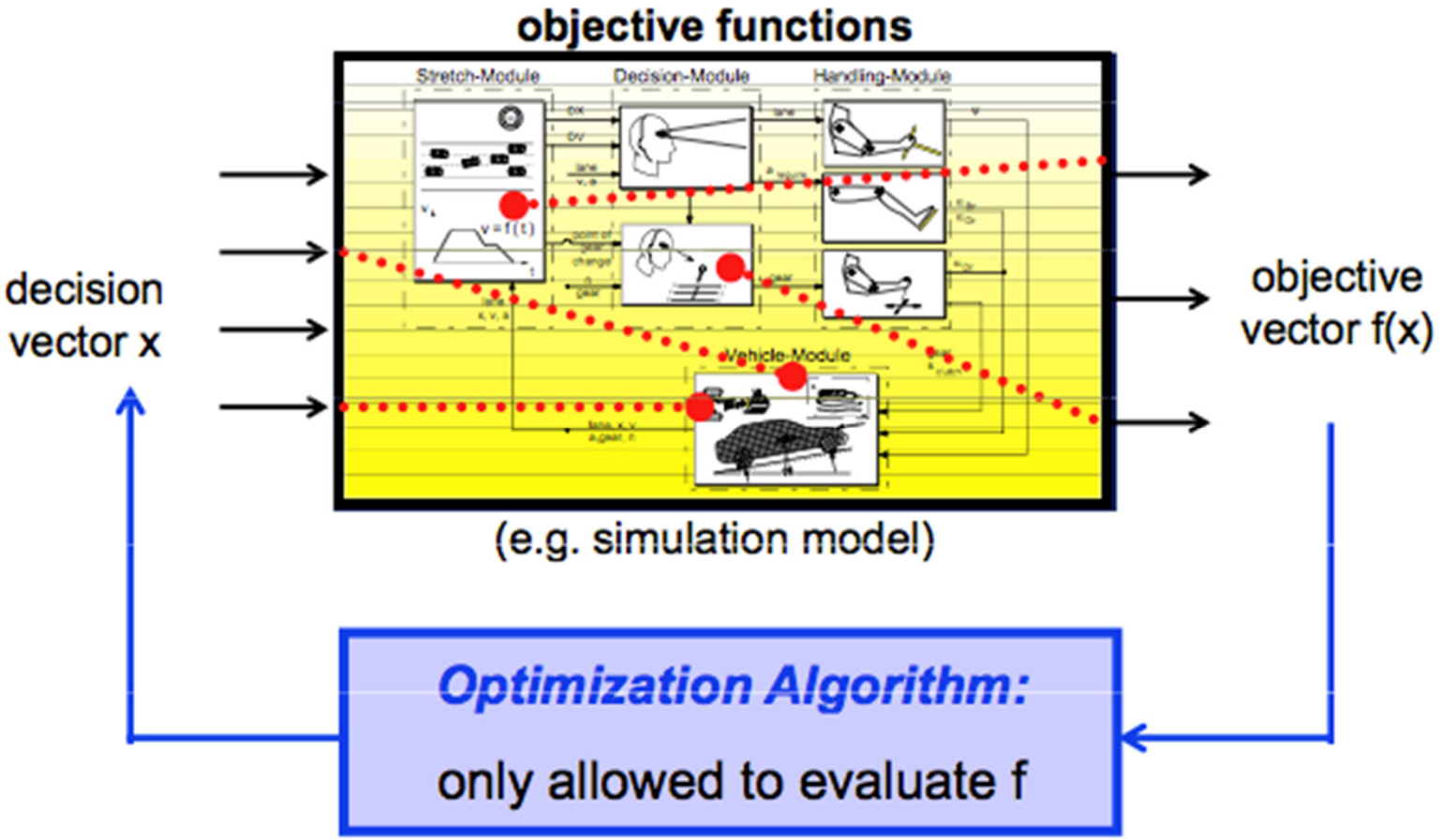
Optimierungsszenario: Beispiel

- Gegeben:
  - Spezifikation der Taskstruktur (Taskmodell)
  - Verschiedene Nutzungsszenarios (Flussmodell)

- Gesucht:
  - Implementierung = Architektur + Abbildung von Tasks + Scheduling
- Ziele:
  - Maximierung der Performance
  - Minimierung der Kosten
- Nebenbedingungen:
  - Speichergröße
  - Latenzen

} (Performancemodell)

Methode: Black-Box-Optimierung



Multikriterielle Optimierung: Grundlegende Definitionen

- Minimierung einer *Zielfunktion* über Vektoren

$$f = (f_1; \ldots ; f_n) : X \rightarrow \mathbb{R}^n$$

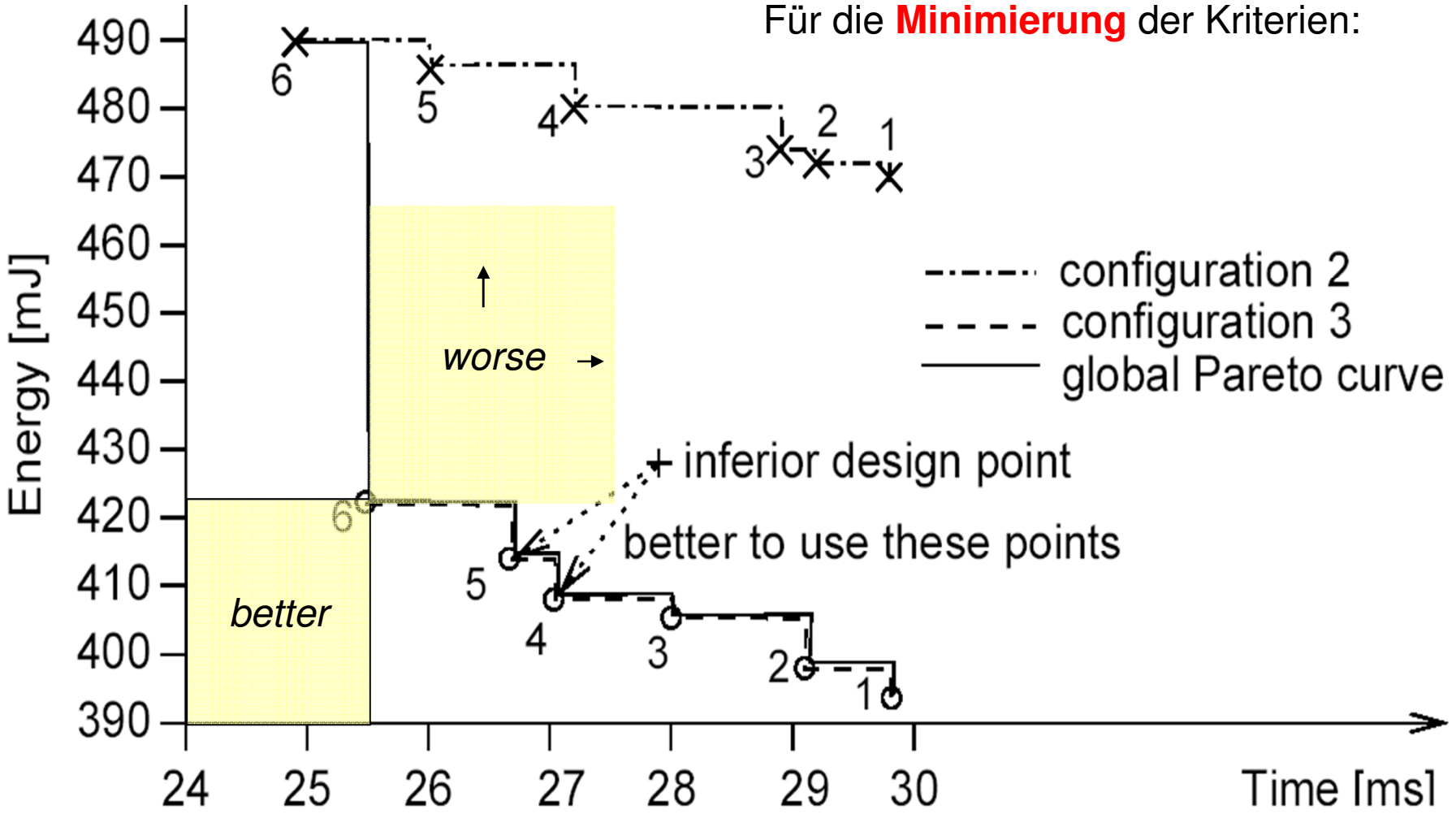
- X bezeichnet den *Entscheidungsraum*, die Menge möglicher Alternativen für das Optimierungsproblem
- Das Bild des Entscheidungsraums X unter der Zielfunktion f wird *Zielraum* $Z \subset \mathbb{R}^n$ genannt mit

$$Z = \{ f(x) \mid x \in X \}$$

- Eine einzelne Alternative $x \in X$ wird auch 'Lösung' genannt, der zugehörige Zielwert $z = f(x) \in Z$ ist der 'Zielvektor'

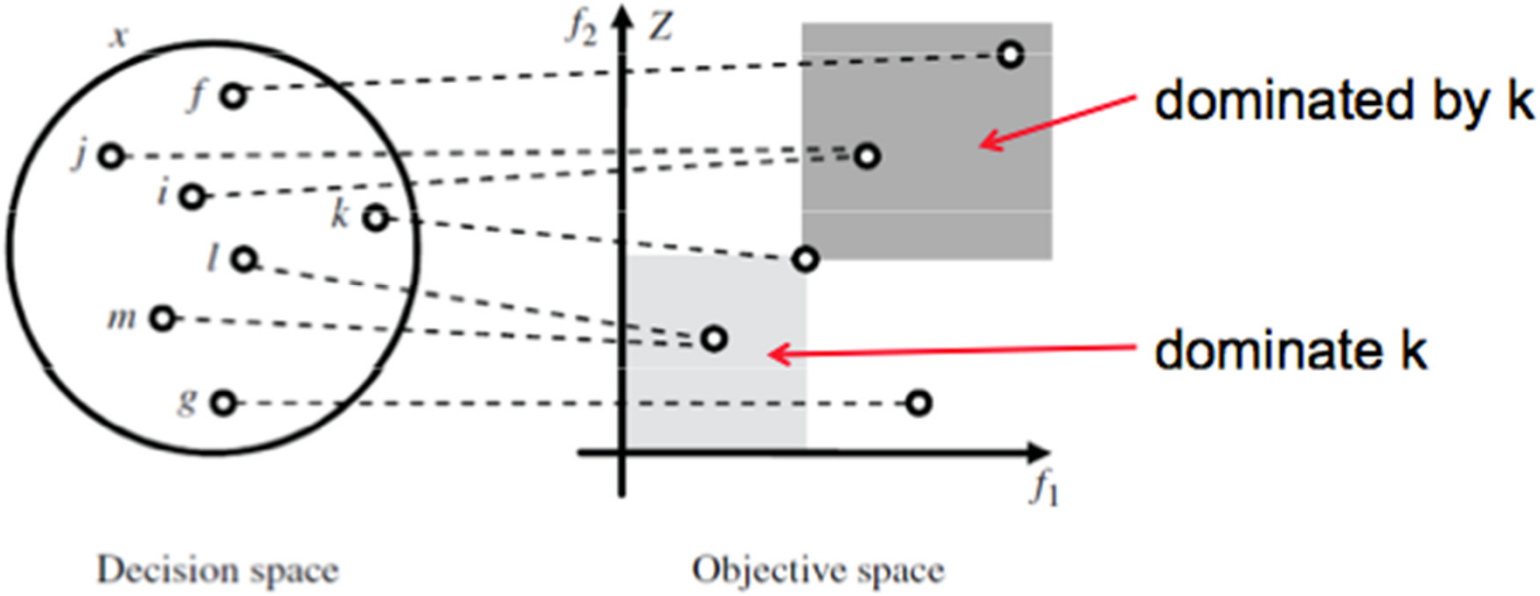
Frage: Wie definiert man das Minimum einer Funktion über Vektoren?

Multikriterielle Optimierung: Pareto-Punkte



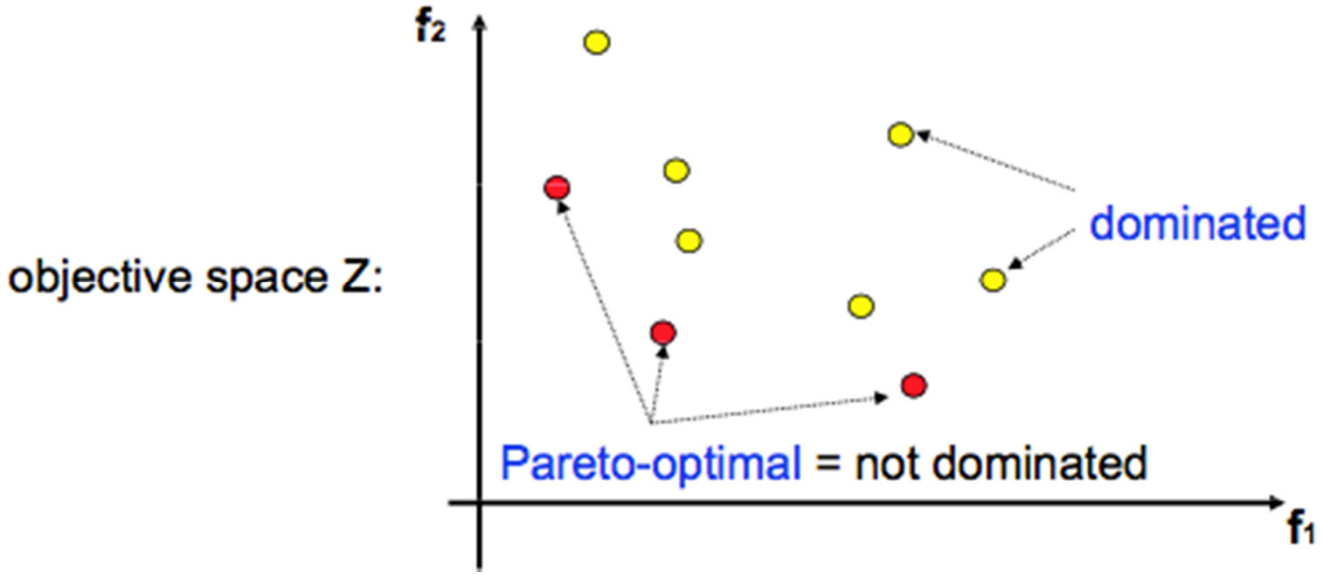
Pareto-Dominanz

- Definition: Eine Lösung a ∈ X ist schwach Pareto-dominant zu einer Lösung b ∈ X (a ⪯ b), wenn sie in allen Zielen mindestens so gut wie diese ist ($f_i(a) \preceq f_i(b)\ \forall\ 1 \leq i \leq n$).
- Eine Lösung a ist besser als b (a ≺ b) gdw. $(a \preceq b) \wedge (b \not\preceq a)$



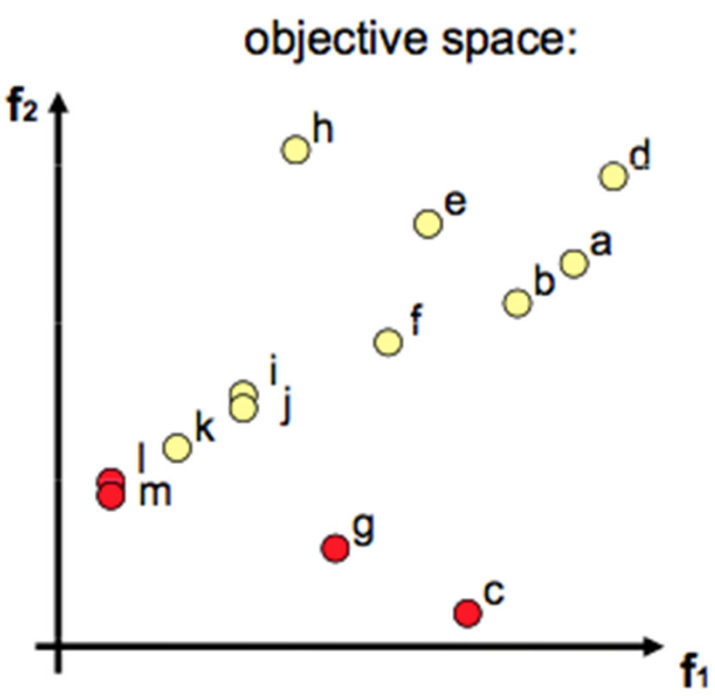
Pareto-optimale Menge

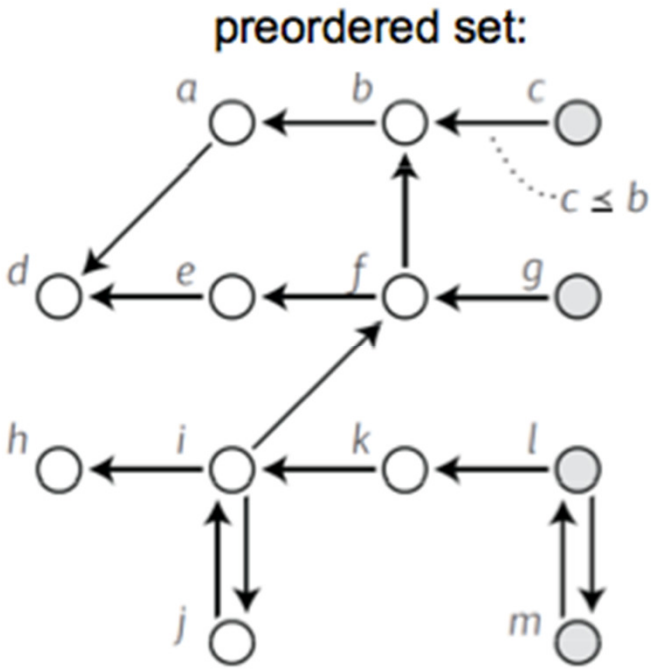
- Eine Lösung heißt *Pareto-optimal*, wenn sie von keiner anderen Lösung in X Pareto-dominiert wird
- Die Menge aller Pareto-optimalen Lösungen heißt die *Pareto-optimale Menge*, ihr Bildbereich im Zielraum die *Pareto-optimale Front* (oder einfach Pareto-Front)



Teilordnung

- Die Dominanzrelation definiert eine *Teilordnung* (preorder) auf allen Entwurfspunkten: Es existiert eine Menge optimaler Lösungen





Optimierungsalternativen

- Verwendung *klassischer Optimierungsverfahren* für einzelne Ziele
  - Simulated annealing, Tabu-Suche
  - Ganzzahlige lineare Programmierung (ILP)
  - Andere konstruktive oder iterative heuristische Methoden
- Entscheidungen werden vor der Optimierung getroffen

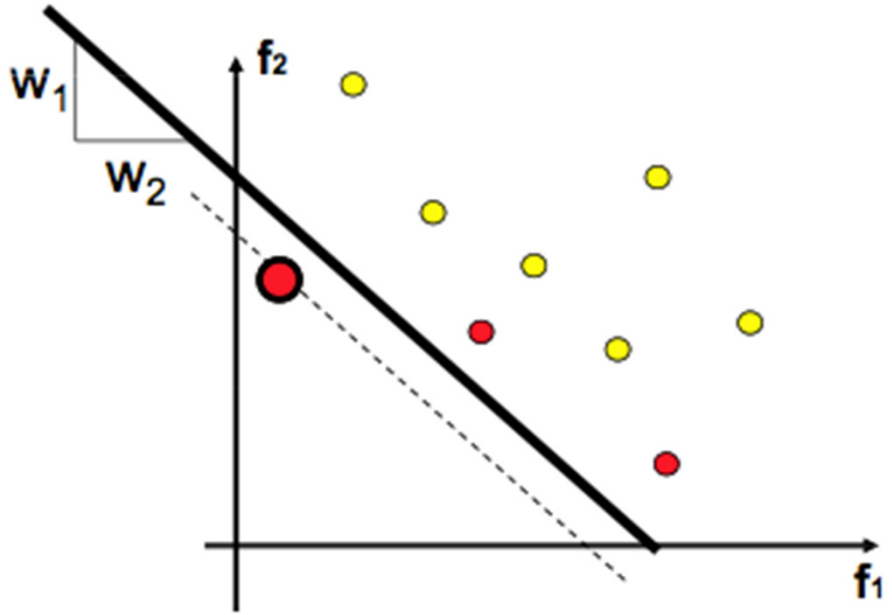
oder

- *Populationsbasierte Optimierungsverfahren*
  - Evolutionäre Algorithmen / Genetische Algorithmen
- Entscheidungen werden *nach* der Optimierung getroffen

Einkriterien-Probleme

- Ein multikriterielles Problem läßt sich auf viele Weisen auf ein Einkriterien-Problem reduzieren
- Beispiel 1: Aggregation (gewichtete Summe):

$$\text{minimiere } f = w_1 f_1 + w_2 f_2 + \ldots + w_n f_n$$



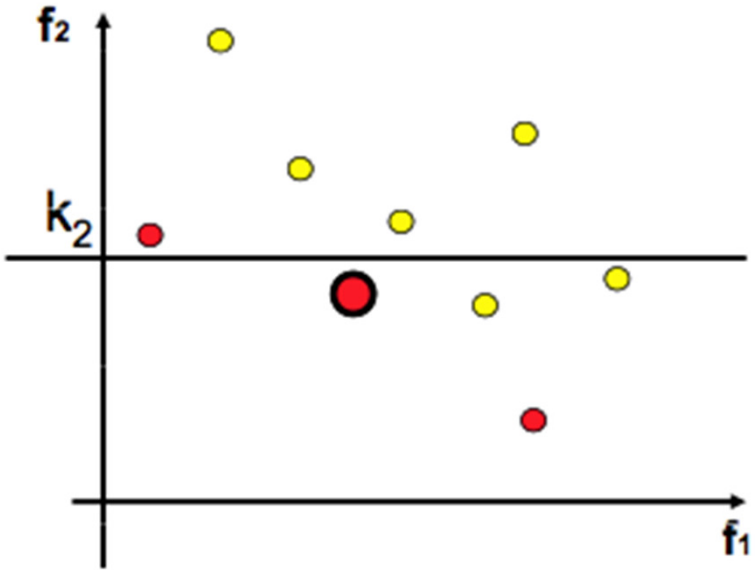
Gewichte müssen vor Optimierung festgelegt werden

Nicht alle Pareto-Punkte werden gefunden

Einkriterien-Probleme

- Beispiel 2: Ein Ziel mit Nebenbedingungen

$$\text{minimiere } f=f_1 \text{ mit } f_2 \le k_2 \ldots f_n \le k_n$$



Grenzen müssen vor Optimierung festgelegt werden

Alle Pareto-Punkte können durch ‘Durchlaufen’ aller Grenzen gefunden werden

Populationsbasierte Optimierung

- Populationsbasierte Verfahren untersuchen *eine Menge an Lösungen* gleichzeitig
- *Evolutionäre Algorithmen* sind black-box Optimierungs-verfahren, die zufallsgesteuert und populationsbasiert sind

- Wie verwandte Methoden (z.B. simulated annealing) gehen sie von der Annahme aus, dass bessere Lösungen meist in der *Nachbarschaft* guter Lösungen zu finden sind
- Lokale Minima werden vermieden durch
  - Randomisierung, z.B. im Nachbarschaftsoperator oder der Selektion besserer Lösungen
  - Lokale und globale Nachbarschafts-Operatoren

Evolutionäre Algorithmen (1)

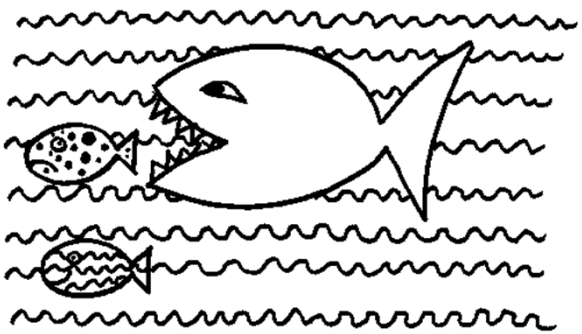
- *Evolutionäre Algorithmen* basieren auf dem kollektiven Lernprozess innerhalb einer Population von Individuen. Jedes Individuum stellt einen Punkt im Raum der optimalen Lösungen zu einem gegebenen Problem dar
- Die Population wird mit beliebigen Werten initialisiert und entwickelt sich hin zu immer besseren Bereichen des Suchraums durch randomisierte Prozesse:
  - *Selektion* (in einigen Algorithmen deterministisch),
  - *Mutation* und
  - *Rekombination* (fehlt komplett in einigen Algorithmen)

[Bäck, Schwefel, 1993]

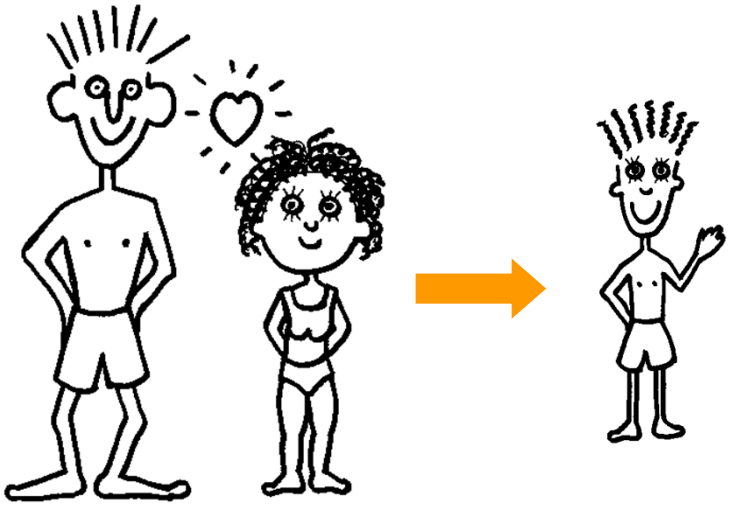
Prinzipien evolutionärer Algorithmen

Prinzipien der Evolution

❶ Selektion



❸ Rekombination



❷ Mutation

Evolutionäre Algorithmen (2)

- Die Umgebung (das gegebene Ziel der Suche) liefert einen *Eignungswert* (**fitness value**) für jeden der Punkte.
- Die Selektion bevorzugt für die Reproduktion Individuen mit größerer Fitness häufiger als solche mit geringerer Fitness
- Der Rekombinationsmechanismus erlaubt es, Information der Vorfahren zu mischen und an die Nachfahren weiterzugeben. Mutation bringt Innovation in die Population ein.

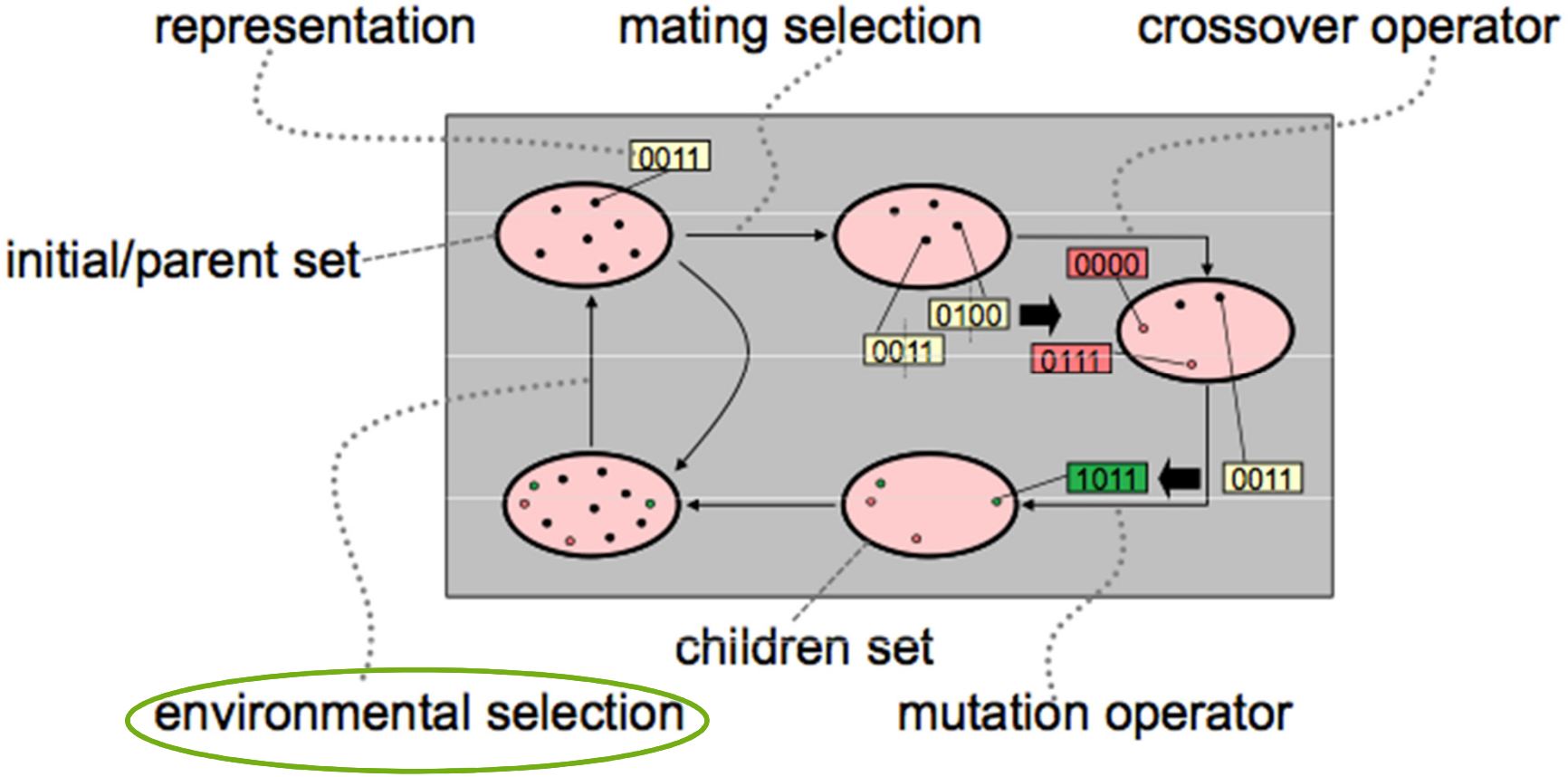
[Bäck, Schwefel, 1993]

Ablauf des evolutionären Algorithmus

(Es existieren viele Variationen der u.a. Vorgehensweise)

1. Eine Menge von Ausganglösungen (Ursprungspopulation) wird (meist zufällig) ausgewählt, die ‘Eltern’-Menge
2. Lösungen der ‘Eltern’-Menge werden selektiert (mating selection) als Grundlage für Schritt 3
3. Lösungen aus Schritt 2 werden durch Nachbarschafts-operatoren verändert, z.B. Kreuzungsoperatoren oder Mutationen. Daraus entsteht die ‘Kind’-Menge
4. Bestimme Vereinigung von ‘Eltern’- und ‘Kind’-Menge
5. Lösungen der Menge aus 4. werden aufgrund ihrer Vorteile ausgewählt, um die neue ‘Eltern’-Menge zu konstruieren (Umgebungsselektion)

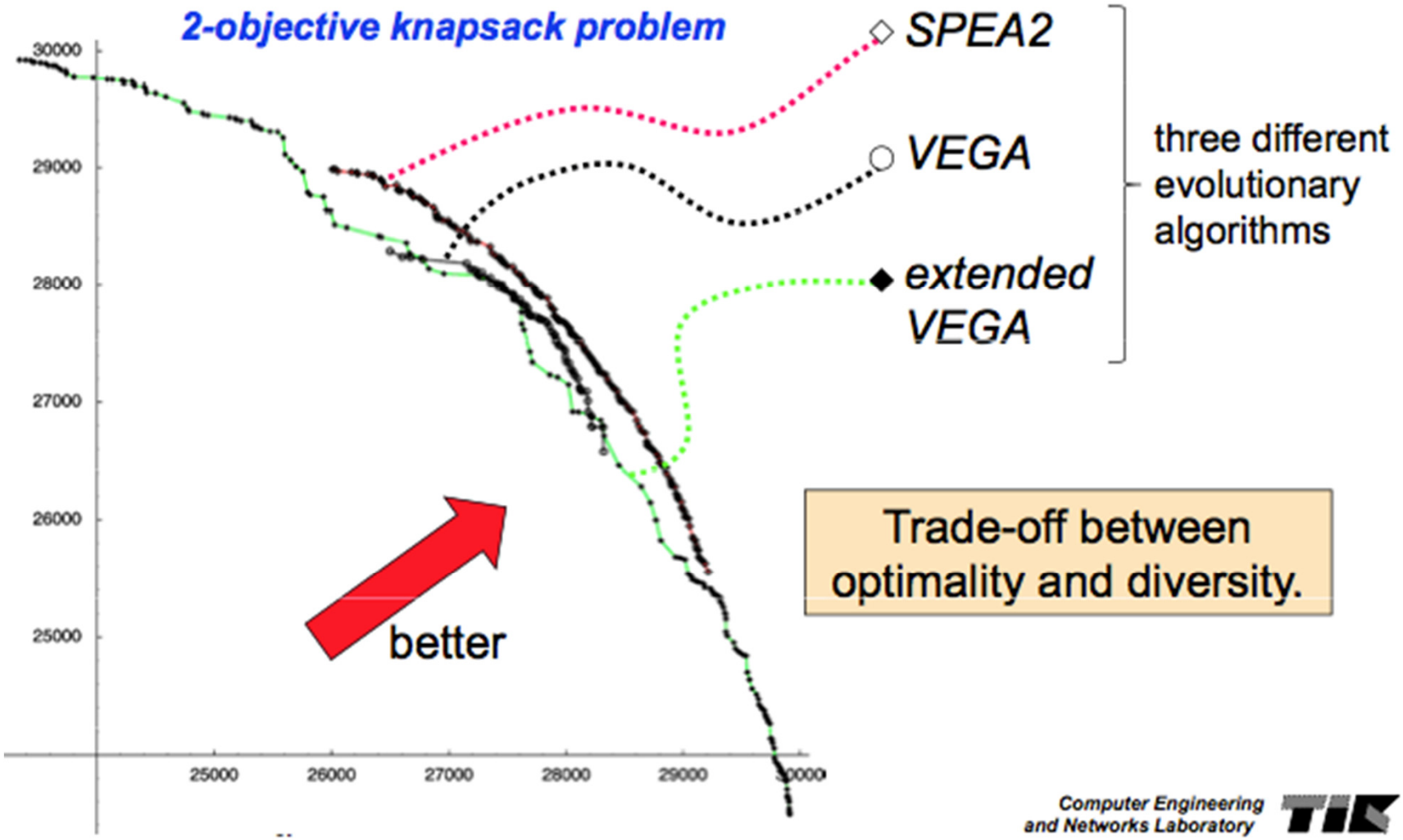
Ablauf des evolutionären Algorithmus



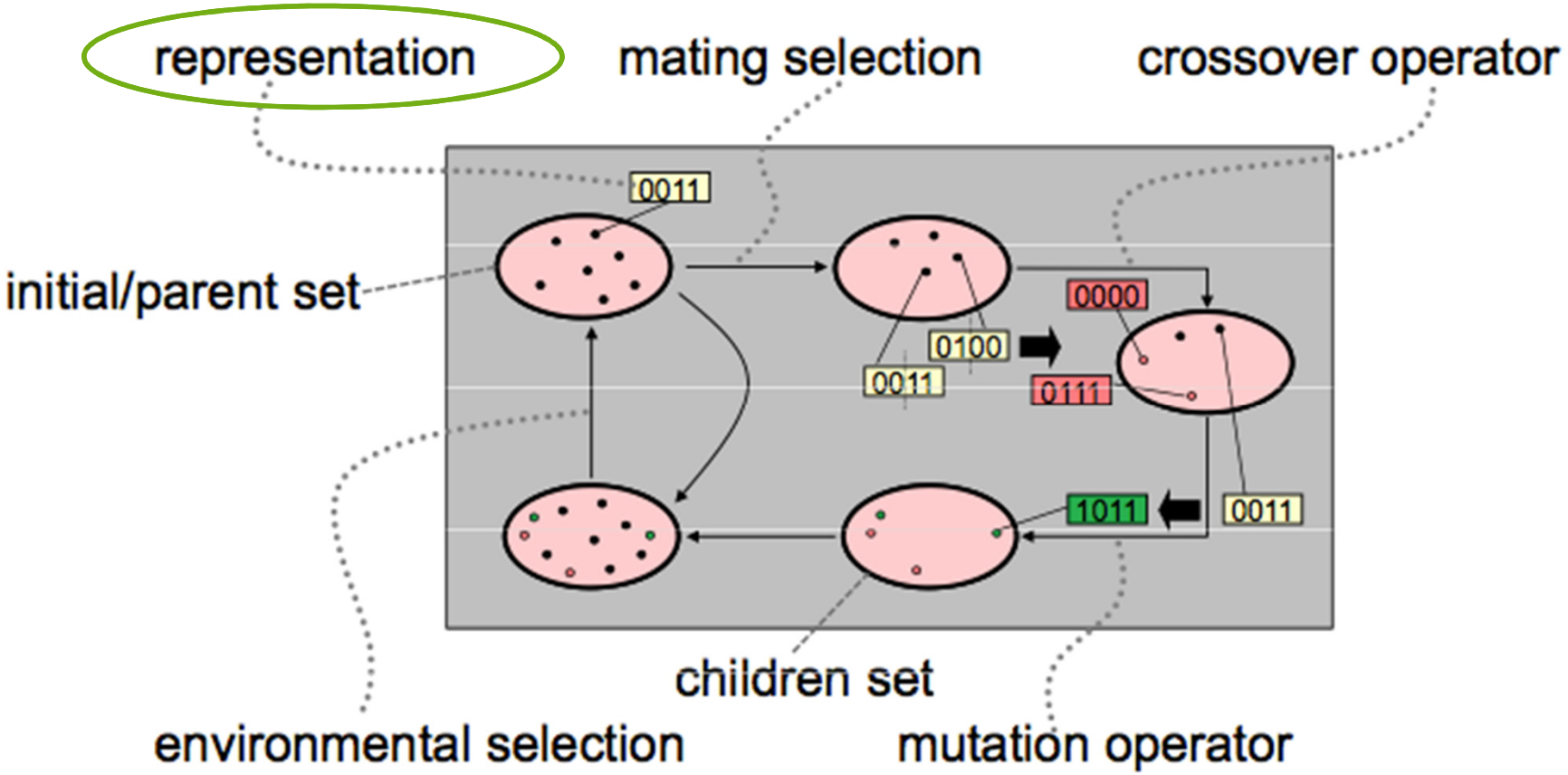
Umgebungsselektion

- Wie wählt man Lösungen aus, die aus der Population entfernt werden sollen?
- *Informelle Kriterien:*
  - Lösungen sollten ‘nahe’ an der (unbekannten) Pareto-optimalen Front liegen (Optimalität)
  - Lösungen sollten große Teile des Zielraums abdecken (Diversität)
- *Grundlegende Idee:*
  - Optimierung von Mengen erfordert Definition eines Indikators, der die Optimalität der gesamten Menge beschreibt
  - Auswahl der optimalen Teilmenge an Lösungen basierend auf diesem Indikator

Optimalität und Diversität

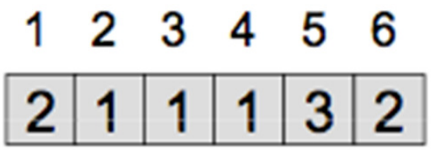


Ablauf des evolutionären Algorithmus

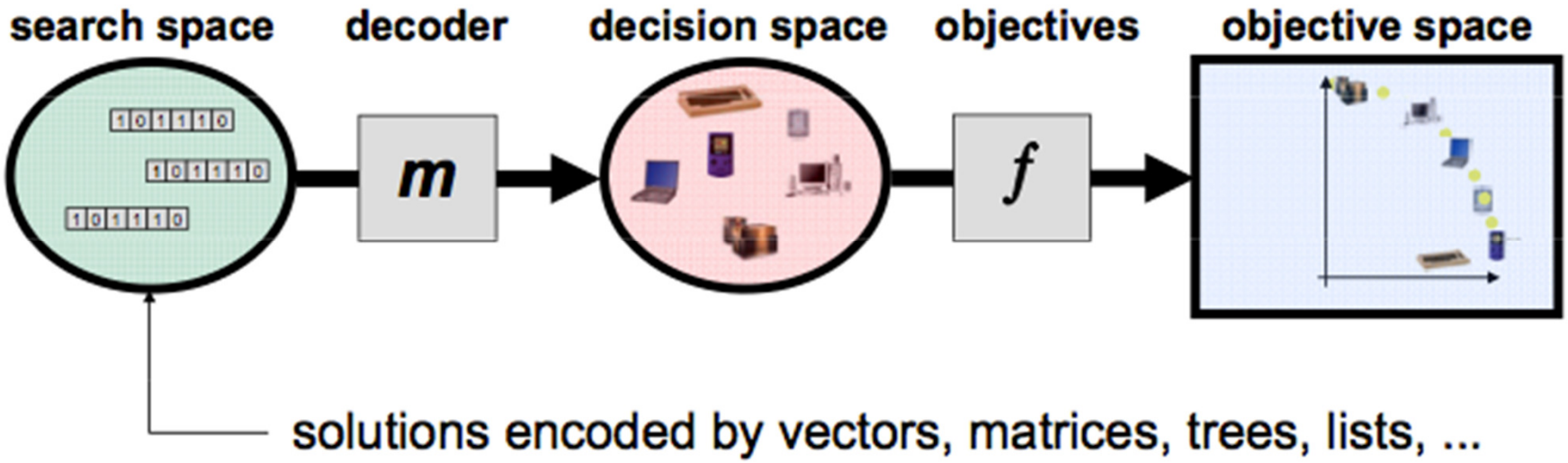


Repräsentation und Nachbarschaft

- Nachbarschaftsoperatoren wie Kreuzung und Mutation basieren meist auf einer Repräsentation (Darstellung) von Lösungen
- Eine Repräsentation entspricht einer abstrakten Datenstruktur, die eine Lösung codiert.
- Nachbarschaftsoperatoren arbeiten auf Repräsentationen
- Einfaches Beispiel:
  - Entscheidungsraum: Alle Partitionierungen von n Objekten in m Blöcke
  - Repräsentation einer einzelnen Lösung durch einen Vektor der Länge n mit Elementen aus [1, m].
  - Z.B. für n=6 und m=3:

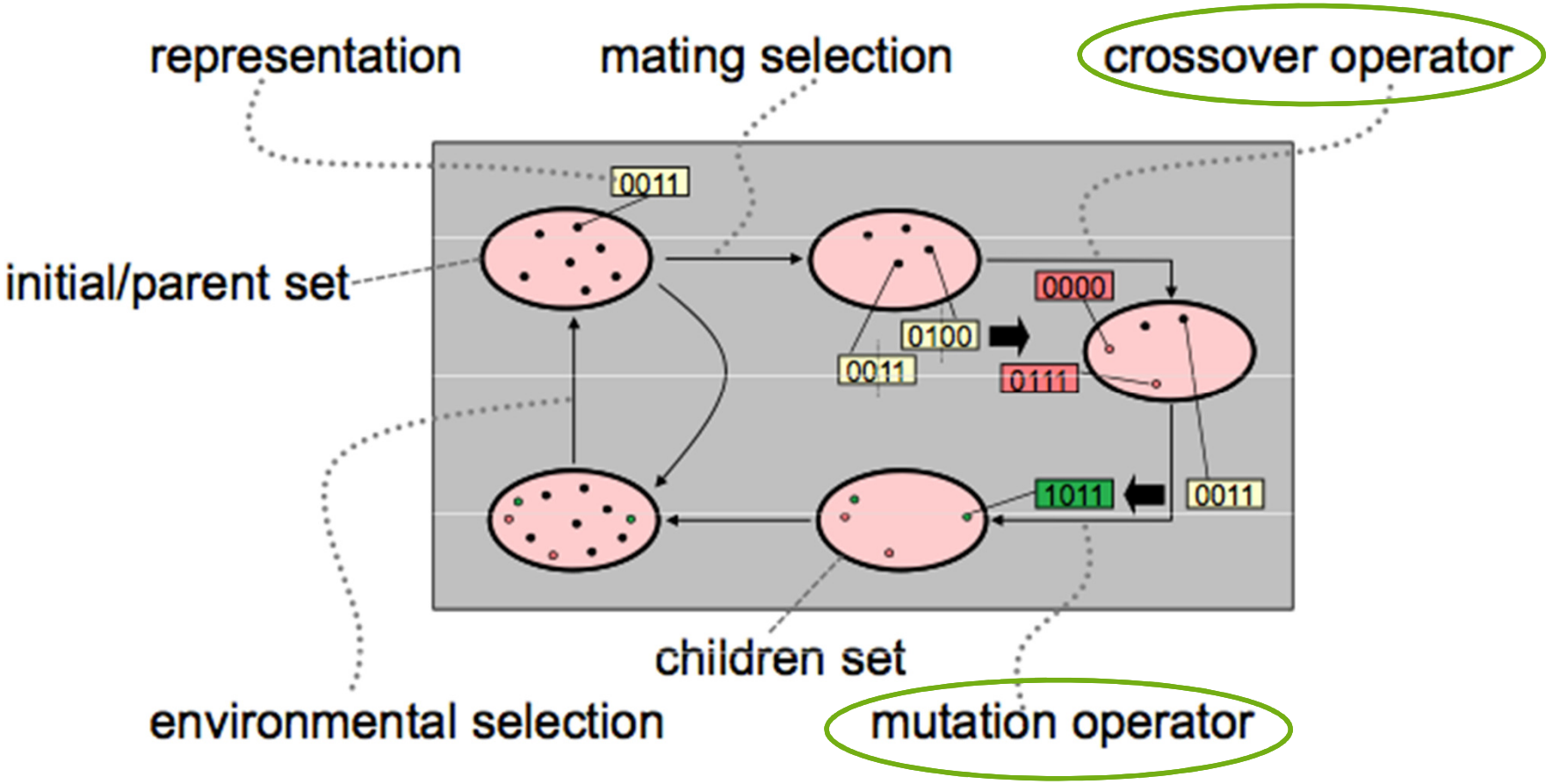


Repräsentation

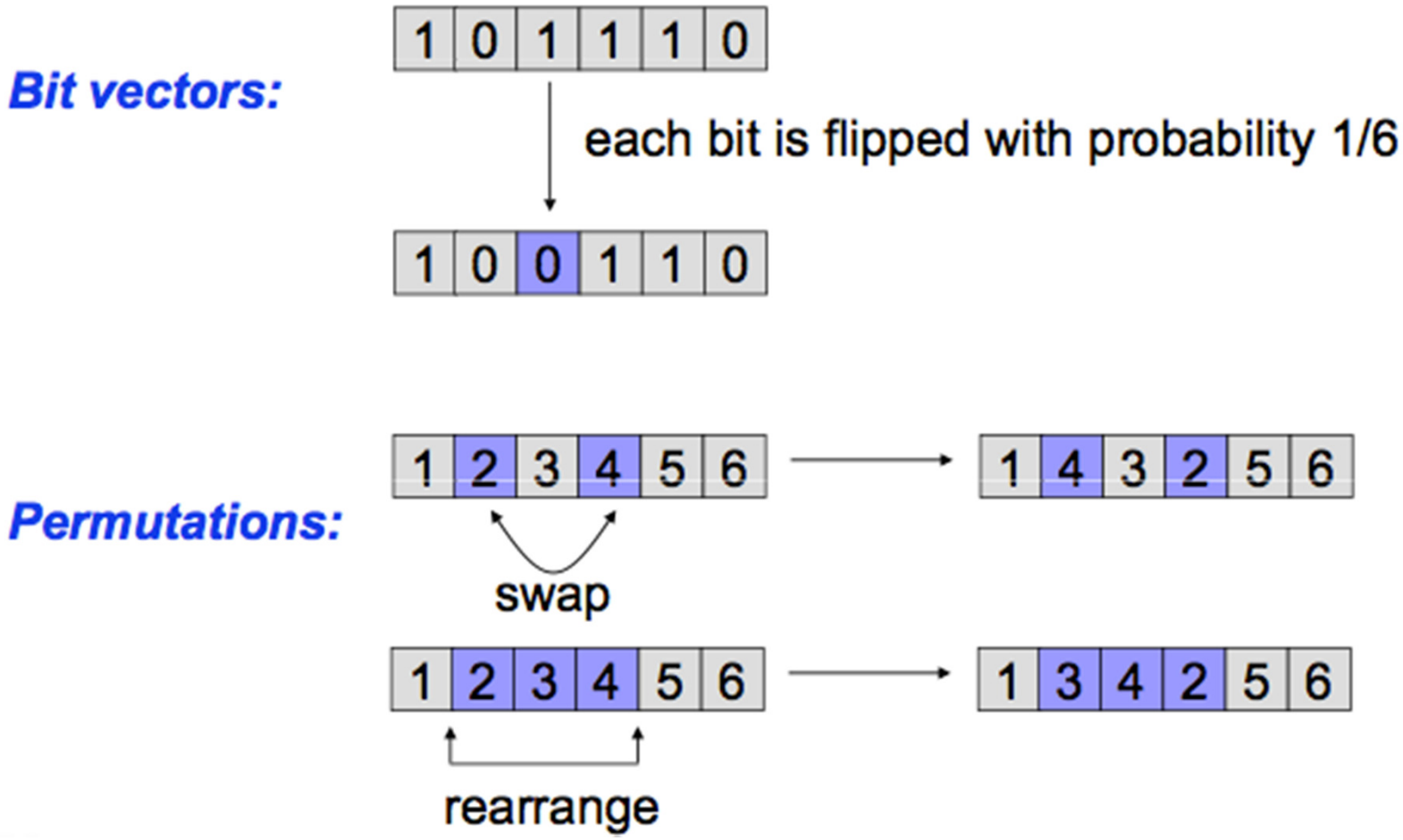


- Kriterien:
  - Vollständigkeit (jede Lösung besitzt eine Codierung)
  - Uniformität (alle Lösungen gleich häufig repräsentiert)
  - Redundanz (Kardinalität des Suchraums im Vergleich zum Lösungsraum)
  - Machbarkeit (jede Codierung bildet auf eine mögliche Lösung ab)

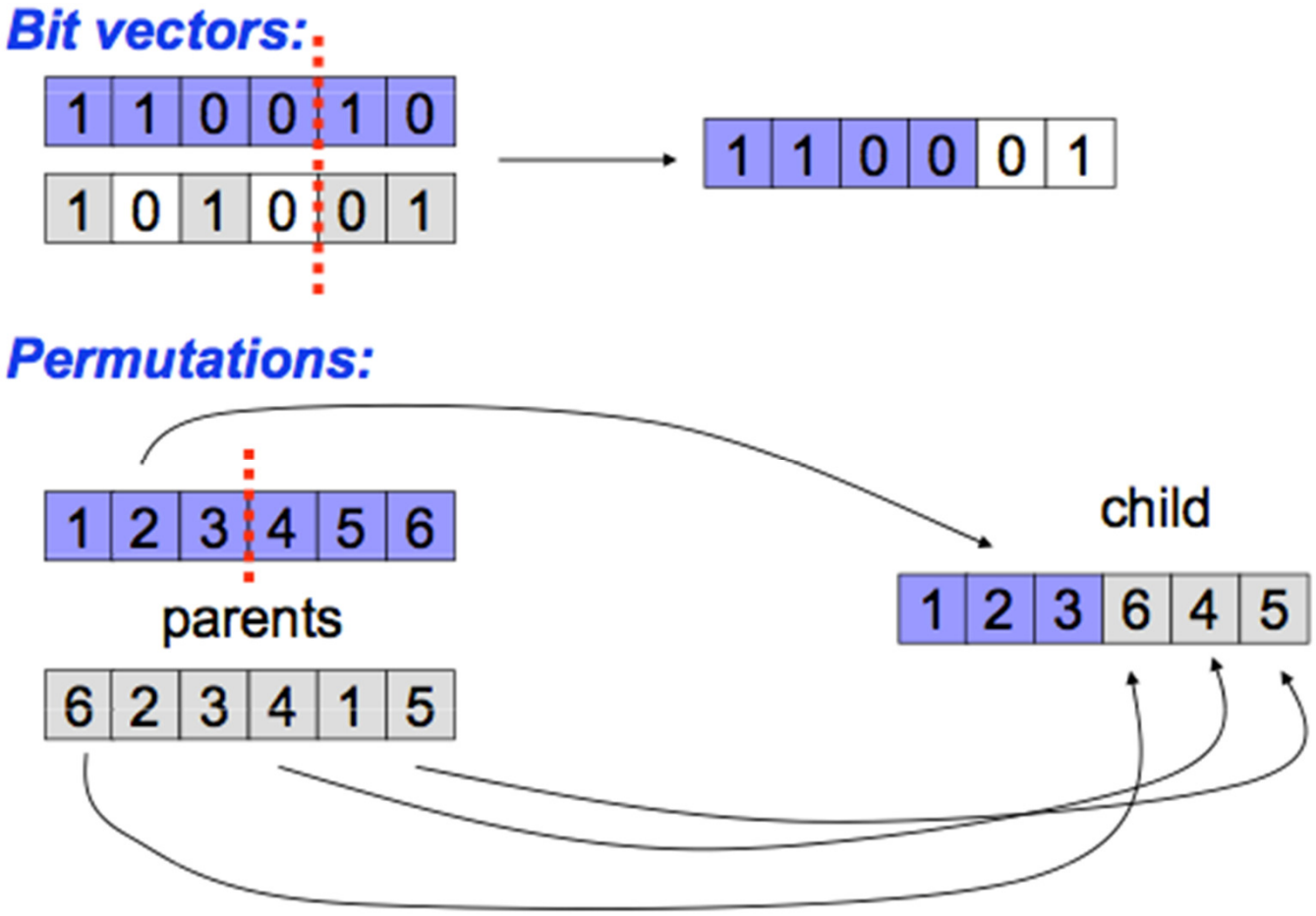
Ablauf des evolutionären Algorithmus



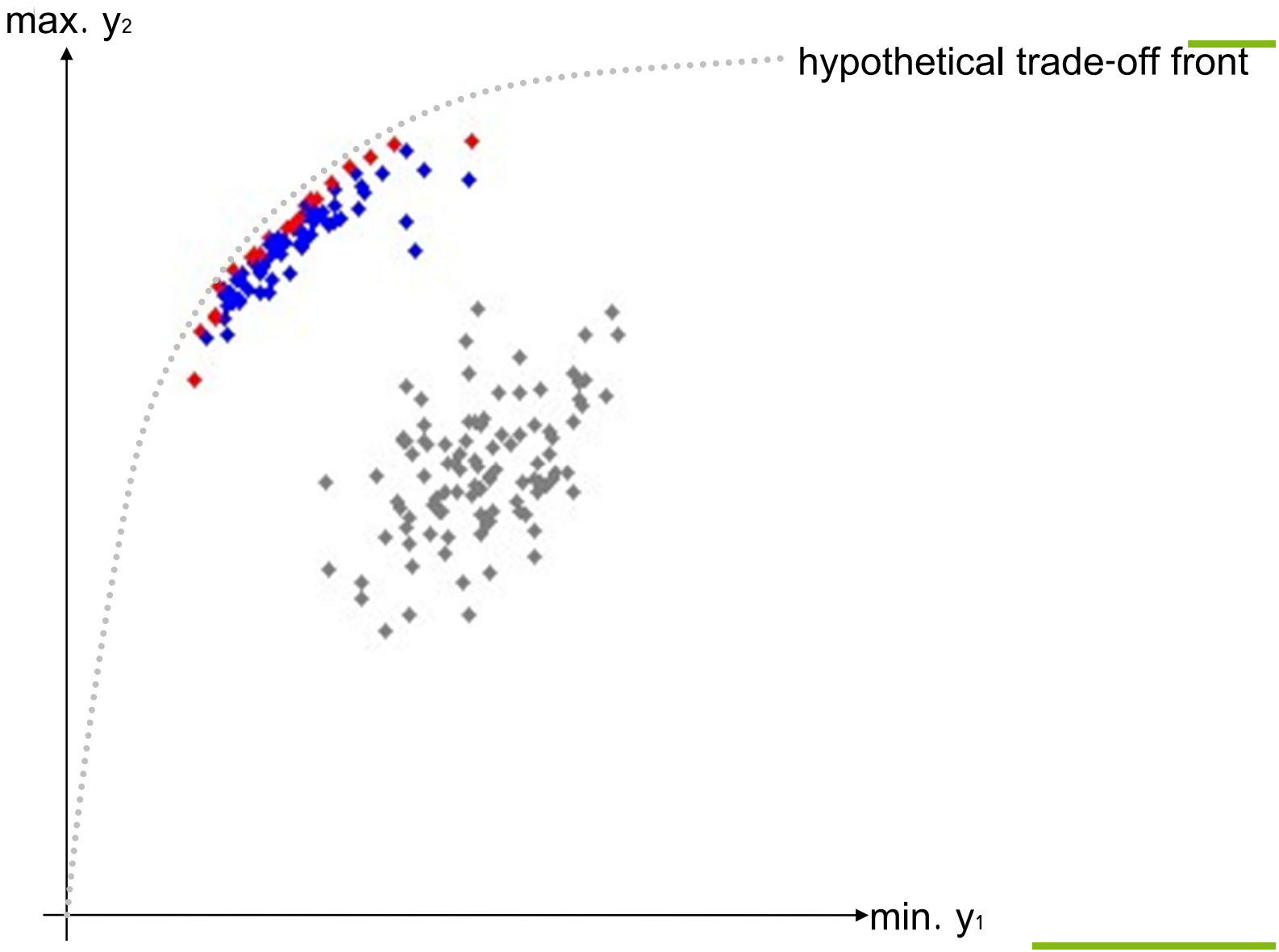
Beispiel: Vektormutation



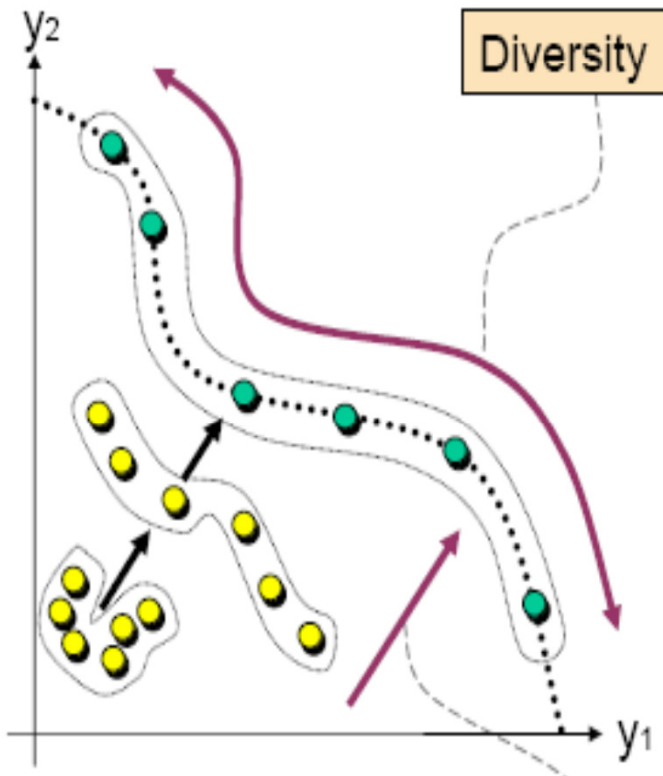
Beispiel: Kreuzung von Vektoren



Ergebnisse eines evolutionären Algorithmus



Issues in Multi-Objective Optimization



Diversity

- How to maintain a diverse Pareto set approximation?

❷ **density estimation**

- How to prevent nondominated solutions from being lost?

❸ **environmental selection**

Convergence

- How to guide the population towards the Pareto set?

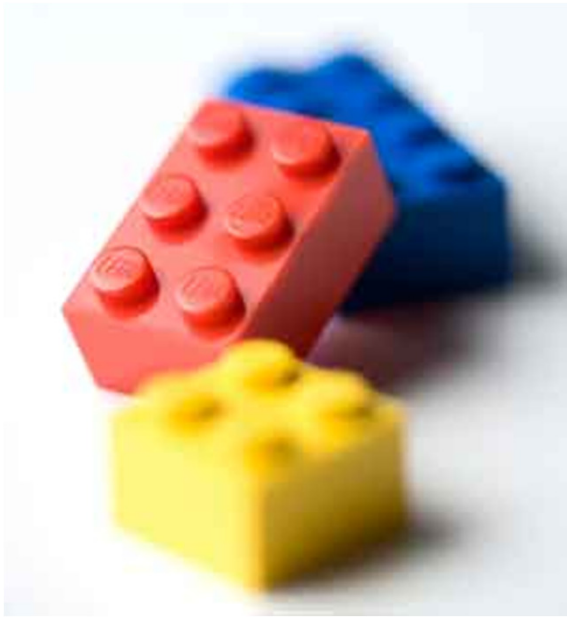
❶ **fitness assignment**

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.

Optimierung mit DOL

- Multikriterielle Optimierung
  - Pareto-Mengen
  - Evolutionäre Algorithmen
- Optimierung in DOL: SPEA2

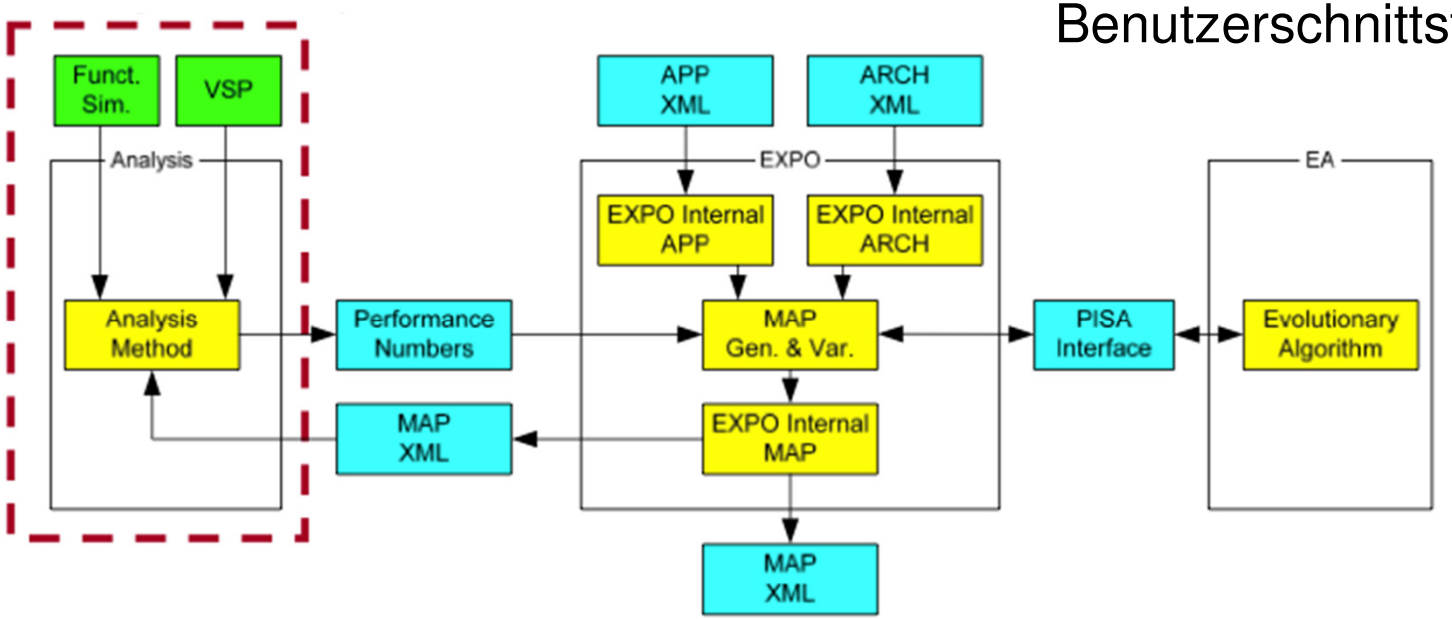


Optimierung der Abbildung

- **Vorgehensweise:**
  - (Einmaliges) Profiling der Anwendung mit funktionaler und instruk-tionsgenauer Simulation
  - Optimierung der Abbildung durch analytisches Modell

Optimierungswerkzeuge:

- SPEA2: Evolutionärer Algorithmus für Optimierung
- EXPO
  - Erzeugung der Abbildung
  - Performancebewertung
  - Grafische Benutzerschnittstelle



Datenquellen für Optimierung

processor c with worst total runtime

number of activations of process p

runtime of process p on processor c

$$obj_1 = \max_{c \in \mathcal{C}} \left\{ \sum_{\forall p \text{ mapped to } c} n(p) \cdot r(p,c) \right\}$$

communication link with worst load

communication request from channel s

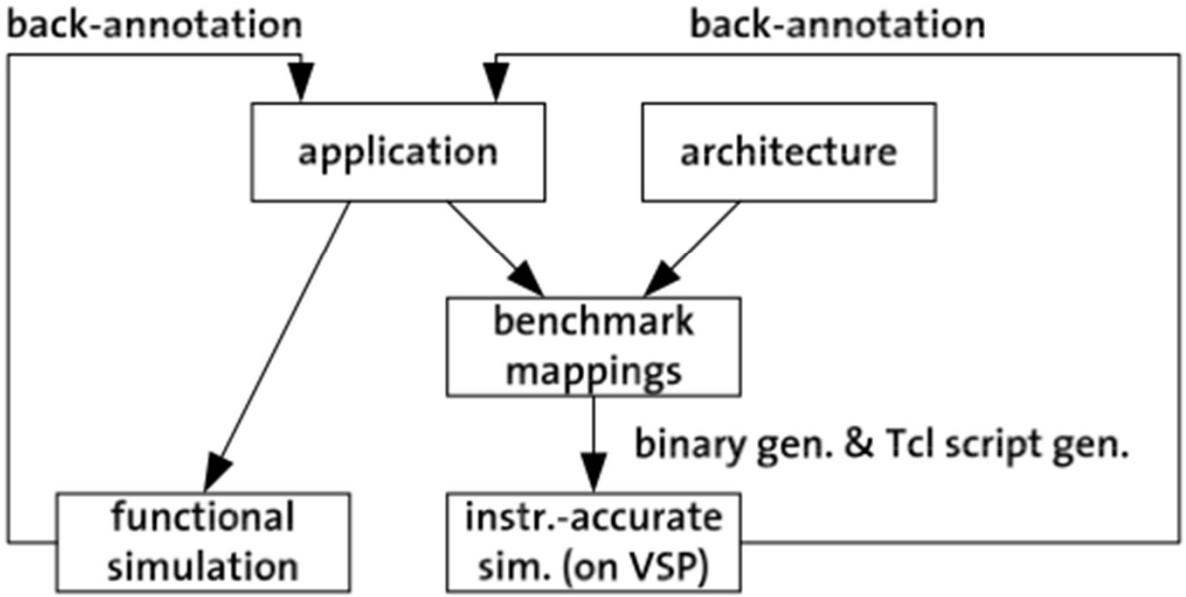
bandwidth of communication link g

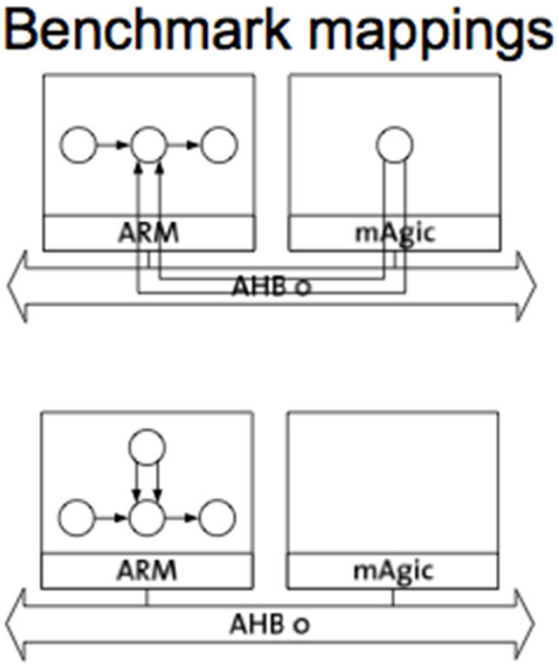
$$obj_2 = \max_{g \in \mathcal{G}} \left\{ \sum_{\forall s \text{ mapped onto } g} \frac{b(s)}{t(g)} \right\}$$

- Statische Parameter:
  - Busbandbreite t(g)
- Funktionale Simulation:
  - Anzahl der Aktivierungen für jeden Prozess n(p)
  - Datenmenge für jeden Kanal b(s)
- Virtual Shapes Platform (VSP):
  - Laufzeit jedes Prozesses auf unterschiedlichen Prozessoren r(p, c) (durch Abbildung von Benchmarks)

Profiling der Ausführung auf VSP

1. Erzeugung der Abbildung des Benchmarks
2. Erzeugung der Binärdateien (mit hardwareabhängiger Software – HdS & Compiler), Erzeugung von Tcl-Skripten
3. Ausführung auf VSP durch Tcl-Script (Erzeugung von Logfiles)
4. Logfile-Analyse und Rückannotation



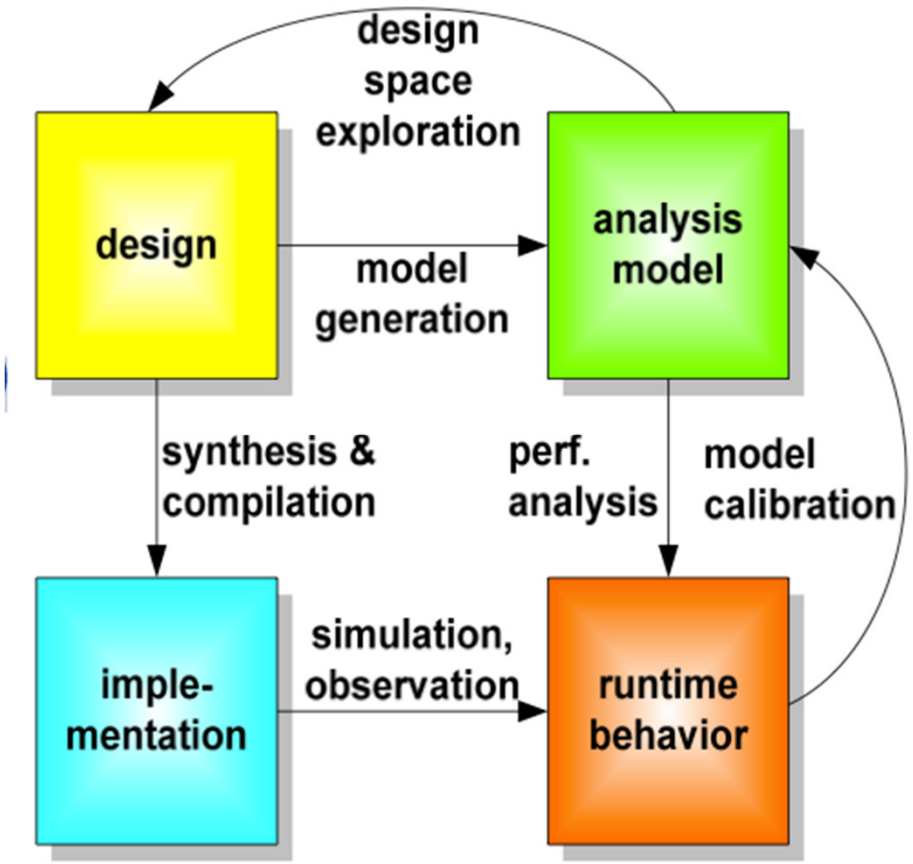


Performance-Analyse

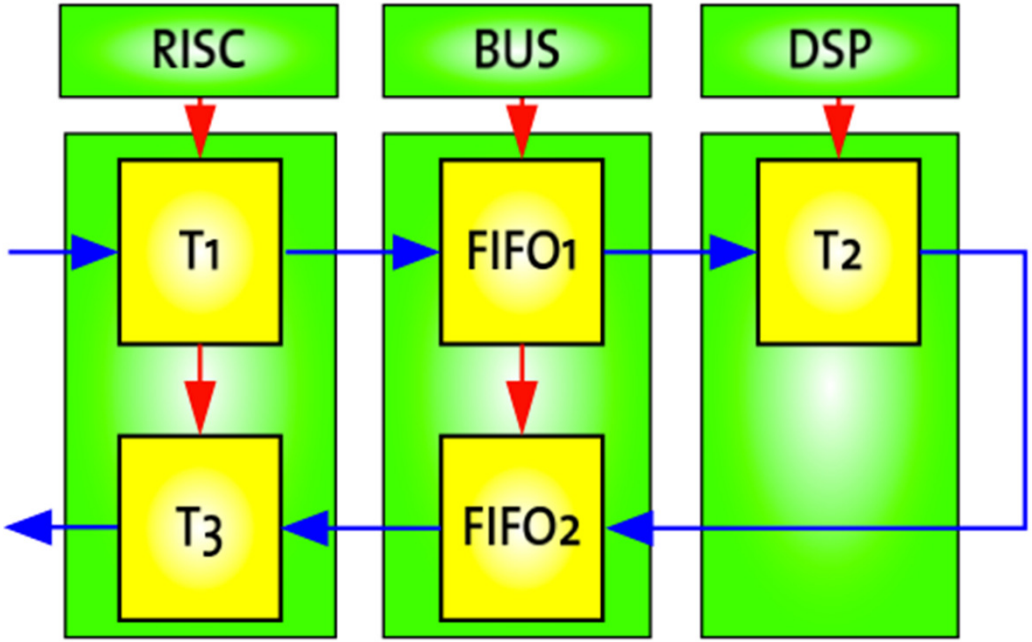
- Ziel
  - Feedback für den Entwickler
  - Verifikation einzelner Entwürfe
  - Entscheidungsgrundlage für Entwurfsraumerkundung
- Kriterien
  - Genauigkeit
  - Geschwindigkeit

DOL Performance-Analyse

- Ziel: Entwurf von *Echtzeit-Systemen* (für Multimedia, Signalverarbeitung)
- Methode: Modular Performance Analysis (MPA)
- Anforderung: Integration von MPA in DOL
  - Erzeugung eines MPA–Modells aus high-level Spezifikation
  - Kalibrierung des MPA-Modells mit genauen Daten



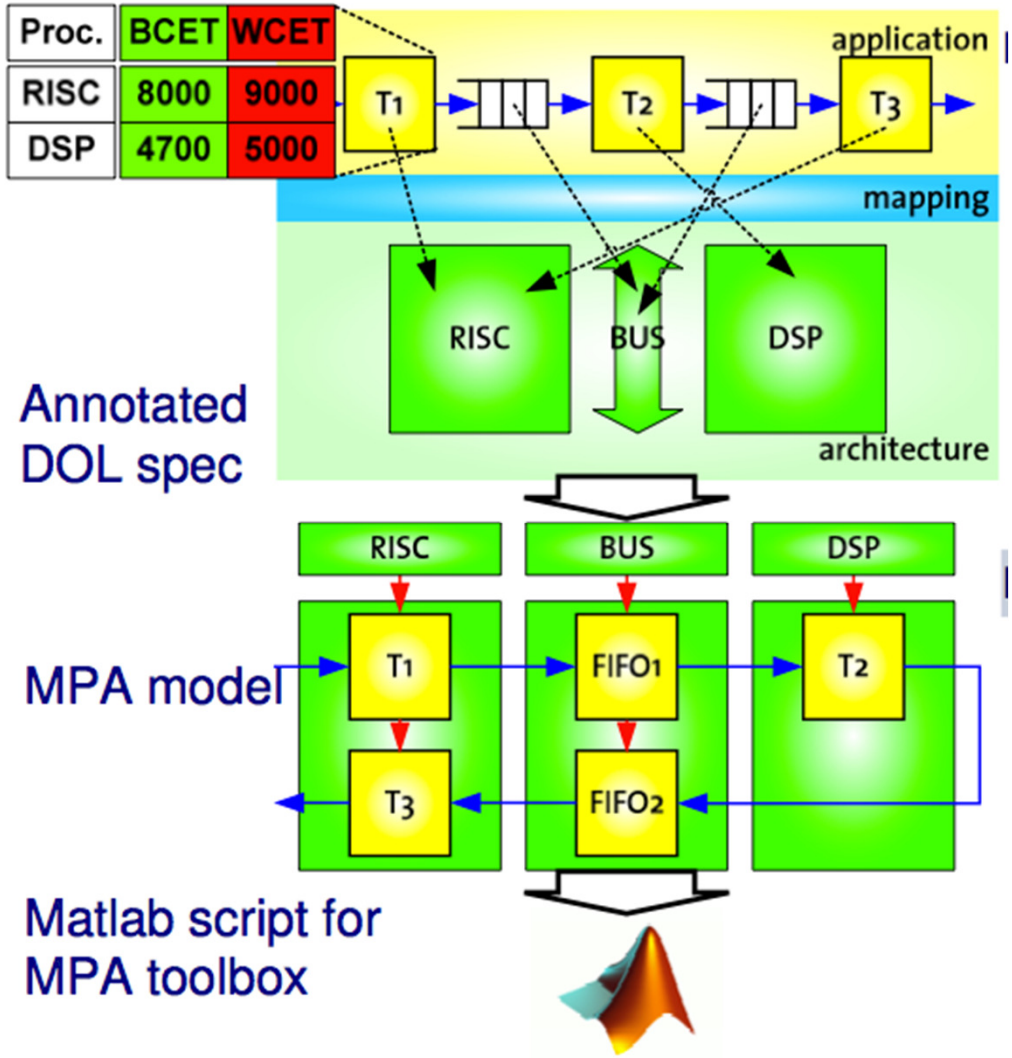
Performance-Analyse mit MPA



- Entwurf: abstrakte Ereignisströme durchlaufen ein Netzwerk abstrakter Prozessoreinheiten
  dynamischer Aspekt: Scheduling
- Ausgabe: worst-case-Grenzen für Systemeigenschaften

- Modular Performance Analysis http://www.mpa.ethz.ch
- Erweiterungen für (große) MPSoCs: komplexe Aktivierungsschemata, Zeitabhängigkeiten, Blockierungs-semantik, zyklische Abhängigkeiten

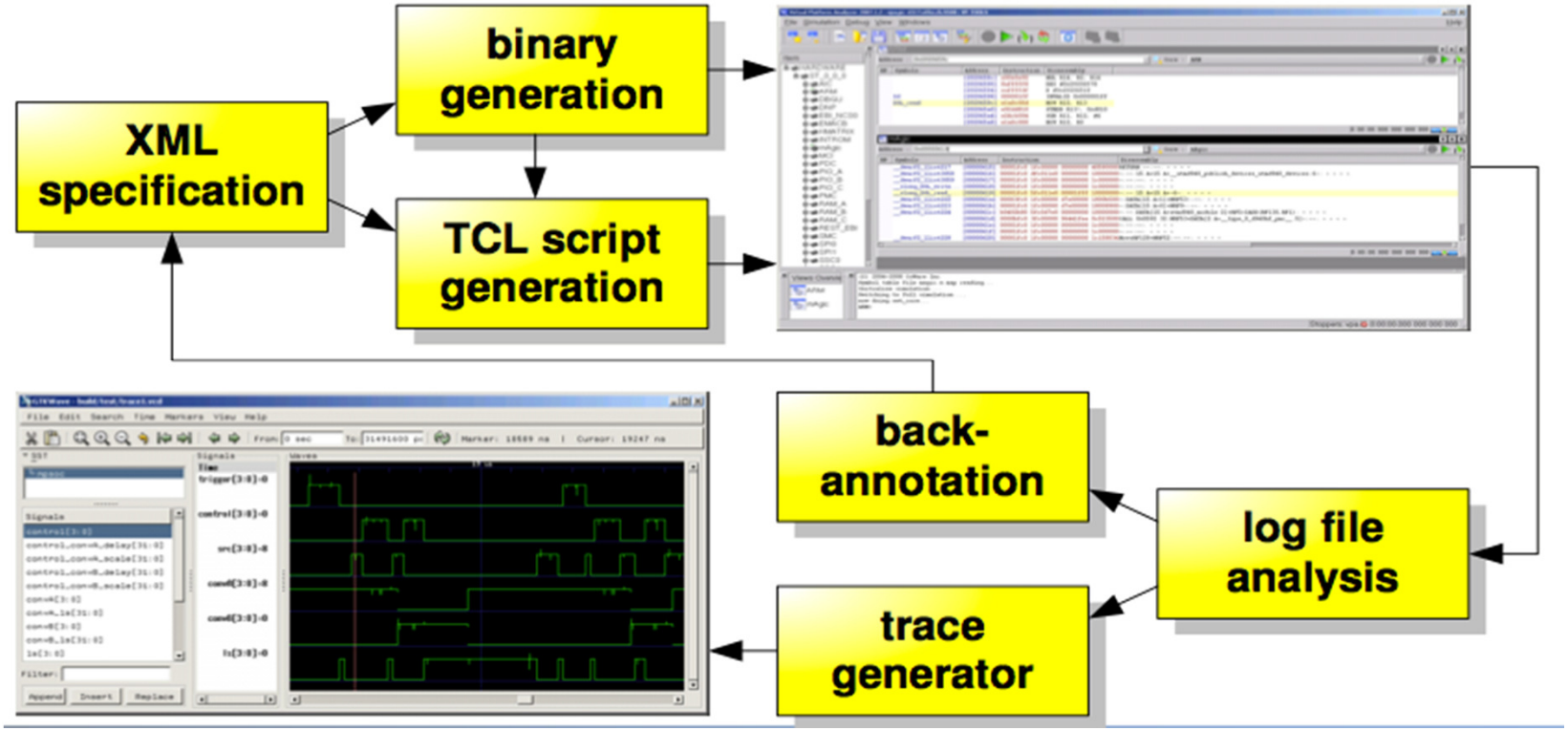
Erzeugung des MPA-Modells



- Automatische MPA Modell erzeugung in 2 Schrittel
  - Framework-unabhängiges Modell(XML-Format)
  - Framework-spezifisches Modell und Matlab-Skript
- Kriterien
  - Relation: DOL Spezif. zu MPA-Modell
  - Sequentielle Evaluation des parall. MPA-Modells
  - Genaue Parameter

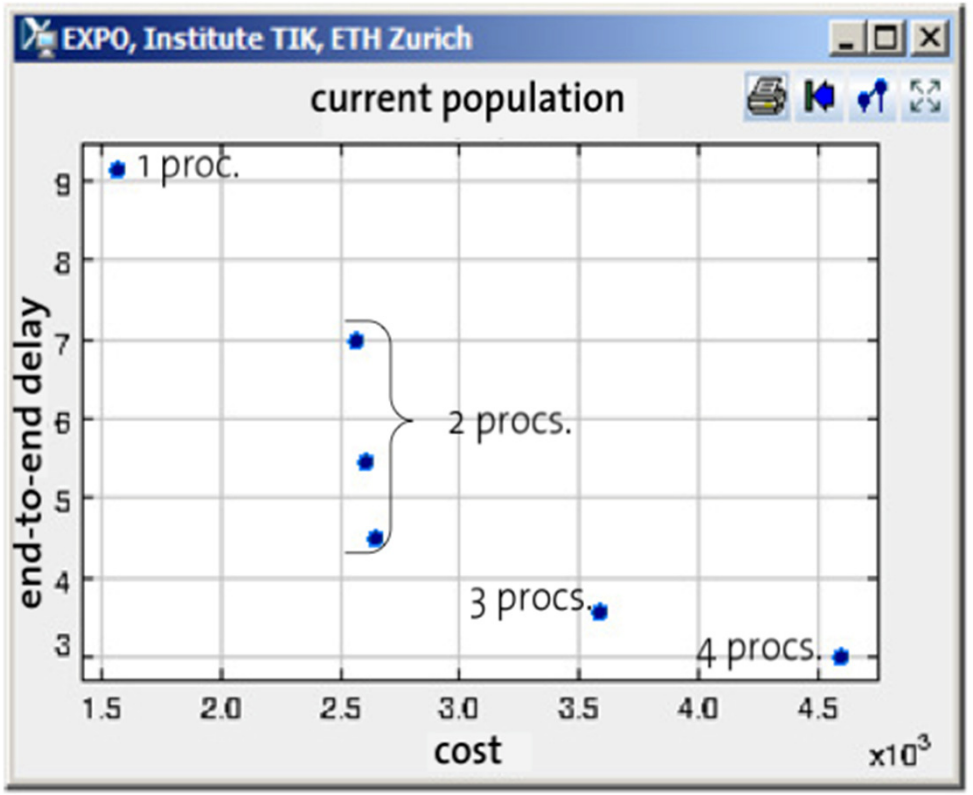
Kalibrierung des MPA-Modells

- Ziel: Sammeln genauer Performance-Daten aus Simulation
- Problem: zu langsam für Entwurfsraumerkundung
- Strategie: vorher Parameter ermitteln durch “Kalibrationsabbildungen”



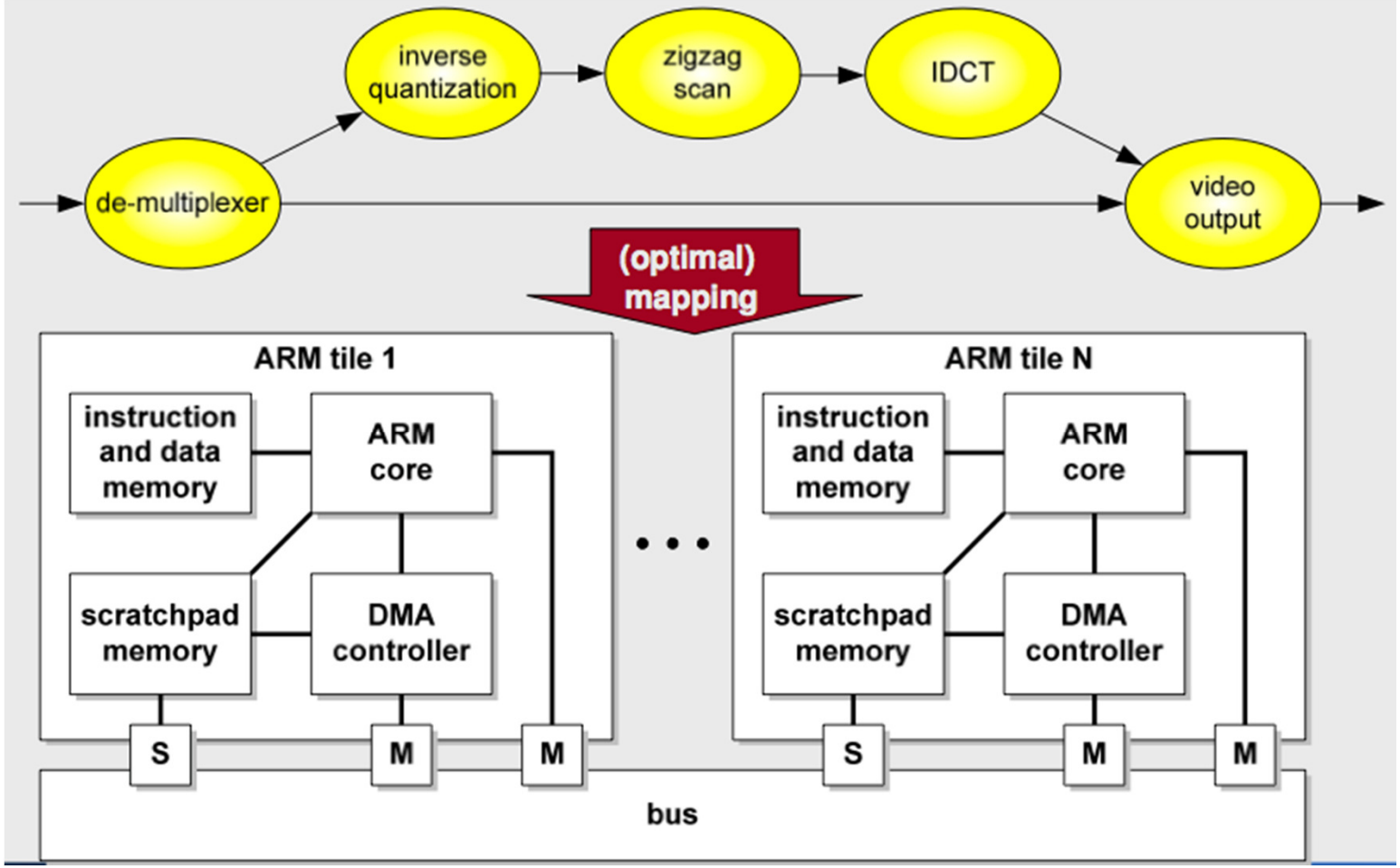
Entwurfsraumerkundung

- PISA/EXPO-Umgebung (Evolutionärer Algorithmus SPEA2)
- Zwei Ziele:
  - Ende-zu-Ende-Verzögerung (obere Grenze in MPA)
- Kosten (additives Modell)
- Population: 60 Individuen
- # Generationen: 50
- Pareto-Front: 12 Lösungen
- Suchzeit: ~2 Stunden



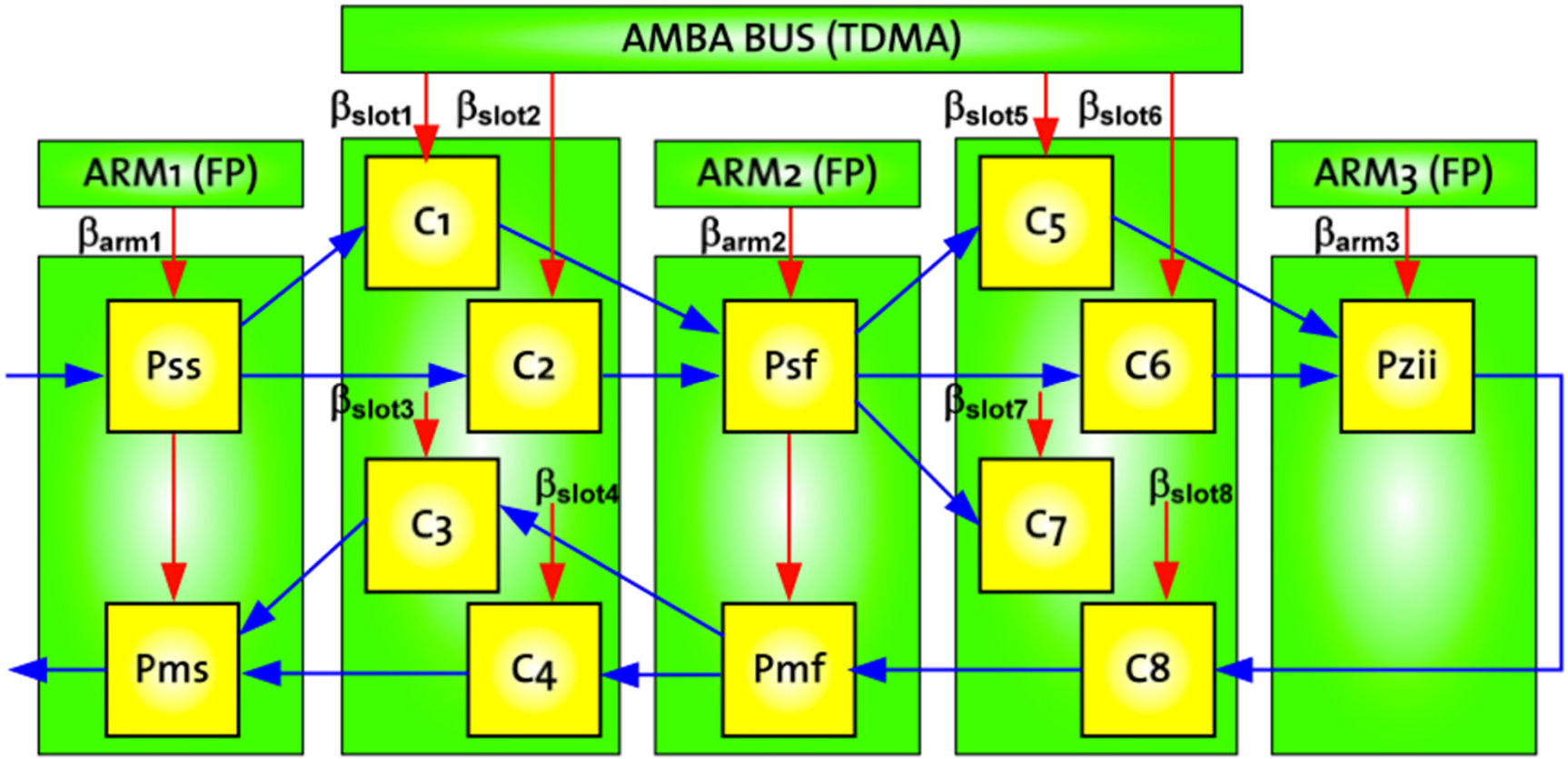
Beispiel

- MJPEG-Anwendung ausgeführt auf MPARM MPSoC



Performance-Analyse

- Abbildung der MJPEG-Anwendung auf ein 3-Kachel MPARM-System



Zeitaufwand

step		duration
model calibration (one-time effort)	functional simulation generation	42 s
	functional simulation	3.6 s
	synthesis (generation of binary)	4 s
	simulation on MPARM	13550 s
	log-file analysis and back-annotation	12 s
model generation		1 s
performance analysis based on generated model		2.5 s

- Modellkalibrierung ist zeitaufwendig
  → nicht in der Entwurfsraumerkundung einsetzbar
- Modellerzeugung und Performanceanalyse in MPA: einige Sekunden → für Entwurfsraumerkundung einsetzbar

Zusammenfassung

- Multikriterielle Optimierung
  - Pareto-Mengen
  - Evolutionäre Algorithmen
- Optimierung in DOL: SPEA2