Systempartitionierung Evol. Algorithmen

Hw-Sw-Co-Design

Übersicht

- Grundlagen Evolutionärer Algorithmen
- Strömungsrichtungen
- Hybride evolutionäre Algorithmen
- Systempartitionierung und -exploration

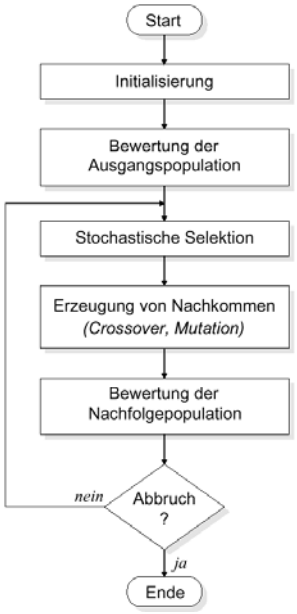
Einleitung

- NP-vollständige Probleme
  - Lösung durch Approximationsverfahren
- Evolutionäre Algorithmen
  - geeignet für große Suchräume
  - breit anwendbares, robustes Verfahren
  - einfaches Grundprinzip
  - geringe Anforderung an Zielfunktion
  - leichte Parallelisierbarkeit
- Historisch bedingt existieren unterschiedliche Strömungsrichtungen

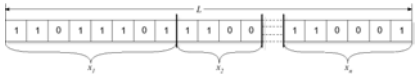
Grundlagen

- Evolutionäre Algorithmen = Algorithmen, die an den natürlichen Optimierungsprozess der Evolution angelehnt sind
- Terminologie:
  - „Individuen“ -> Strukturen (Lösungen)
    - „Genotyp“ -> codierte Lösung (nur bei GA)
    - „Phänotyp“ -> decodierte Lösung (nur bei GA)
  - „Population“ -> Menge von Individuen
  - „Fitness“ -> Güte eines Individuums

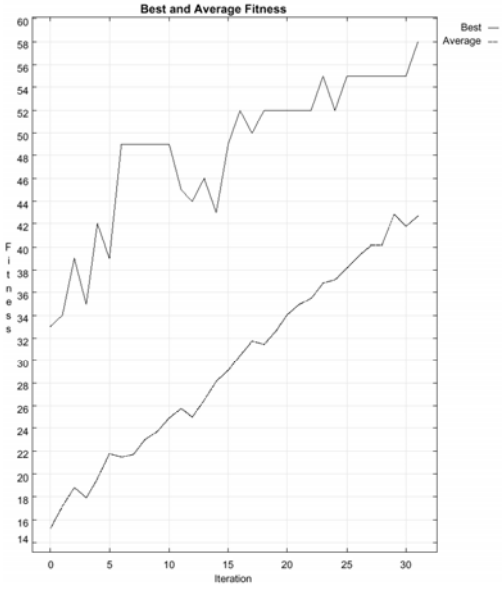
Grundlagen



- Evolutionäre Operatoren:
  - Selektion („survival of the fittest“), z.B. Wettkampfselektion
  - Rekombination (Crossover)
  - Mutation
- Beispielcodierung (GA) eines Individuums:



Grundlagen



Übersicht

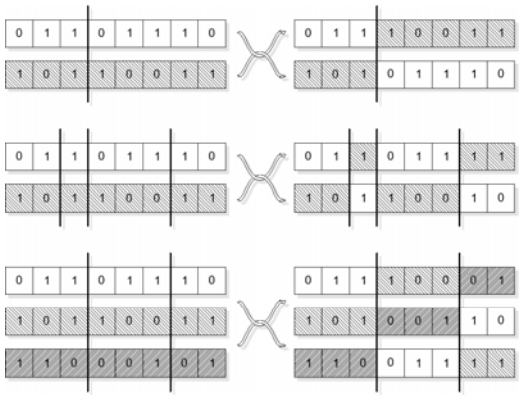
- Grundlagen Evolutionärer Algorithmen
- Strömungsrichtungen
- Hybride evolutionäre Algorithmen
- Pareto-basierte Ansätze

Historische Strömungsrichtungen

- Genetische Algorithmen (GA) (Holland, Goldberg, 60er Jahre)
  - Bitstrings
- Genetische Programmierung (GP) (Koza92)
  - Baumstrukturen
- Evolutionsstrategien (ES) (Rechenberg, Schwefel)
  - reelle Vektoren, deterministische Selektion
- Evolutionäre Programmierung (EP) (Fogel, Owens, Walsh 66)
  - reelle Vektoren, stoch. Selektion, kein Crossover

Grundlagen

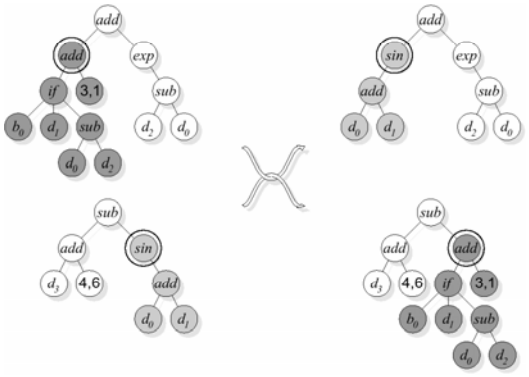
- Rekombination am Beispiel Genetischer Algorithmen (GA):



- Mutation (GA): stochastische Invertierung einzelner Bits

Grundlagen

- Rekombination am Beispiel Genetischer Programmierung (GP):

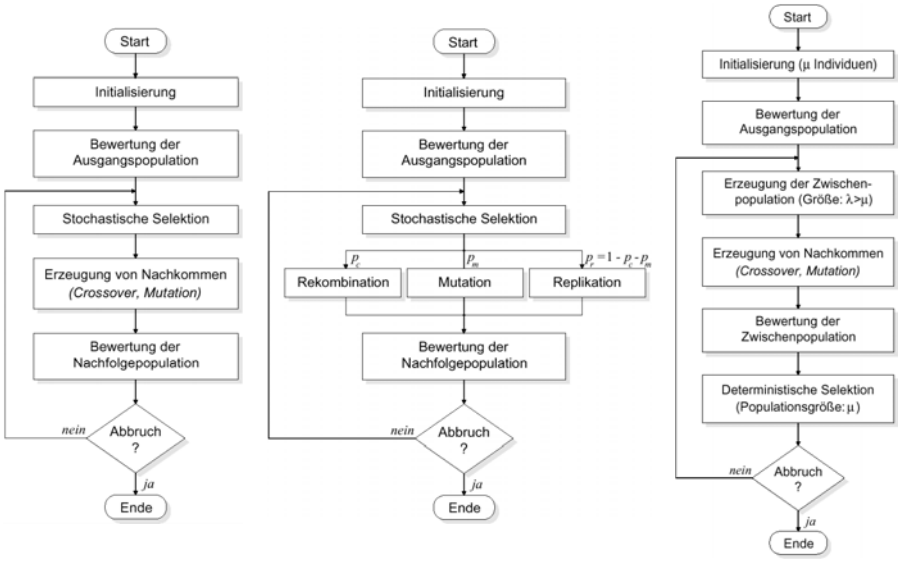


- Mutation (GP): stochastisches Erzeugen von Teilbäumen

Grundlagen

- Rekombination bei Evolutionsstrategien (ES):
  - diskrete Rekombination: stochastischer Austausch der Werte von Variablen zweier Eltern.
  - intermediäre Rekombination: Seien $x_{1,j}$ und $x_{2,i}$, i = 1,...,n die Werte der i-ten Variablen von Elter1 bzw. Elter2, dann gilt: $x_i = u_i\, x_{1,j} + (1 - u_i)\, x_{2,i}$
    U_i: gleichverteilte Zufallsvariable im Intervall [0,1)
- Mutation (ES): Erfolgt bei ES über Mutationsschrittweiten $\sigma_{k,}$die zuerst mutiert werden. Dann: $x'_i = x_j + \sigma_j\, N(0,1)$.

Strömungsrichtungen EA



Übersicht

- Grundlagen Evolutionärer Algorithmen
- Strömungsrichtungen
- Hybride evolutionäre Algorithmen
- Systempartitionierung und -exploration

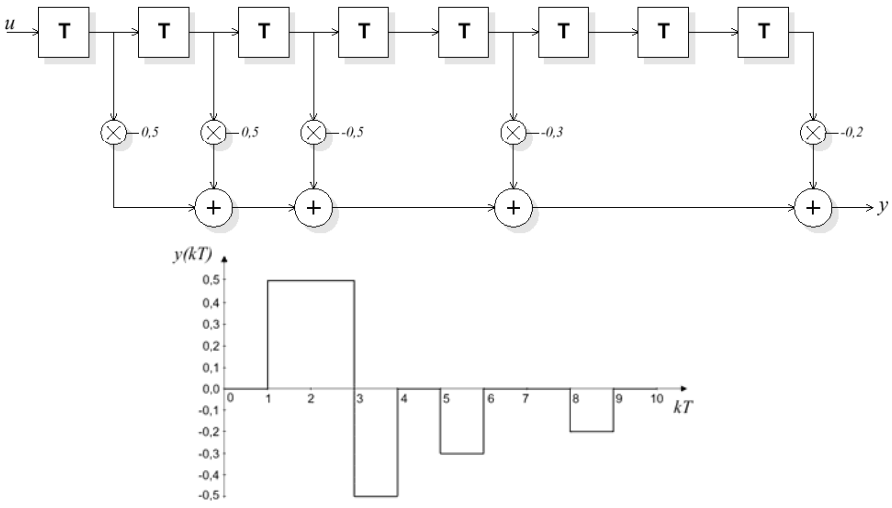
Hybrider Ansatz

- Algorithmische Ebene:
  - abstrakte Beschreibung Evolutionärer Algorithmen auf der Basis von Individuen, Populationen und evolutionären Operationen
  - Interfaces für Individuen und Operatoren wie Selektion, Rekombination und Mutation
  - Populationsklasse; innerhalb dieser Klasse: Implementierung von Selektionsoperatoren
- Datenrepräsentationsebene:
  - problemspezifische Verfeinerung der Lösungsdarstellung und der zugehörigen Operationen

Hybrider Ansatz

- Teil-Implementierung von Individuenklassen:
  - Bit-, Ganzzahl, Realzahl-, Permutationsvektoren und Baumstrukturen
- Implementierung bekannter Rekombinations- und Mutationsoperatoren innerhalb dieser Klassen
- Zusätzlich: Unterstützung für Mehrzieloptimierung: „Strength-Pareto-EA“
- Gruppierung unterschiedlicher Datenrepräsentationen: „Hybride Strukturen“

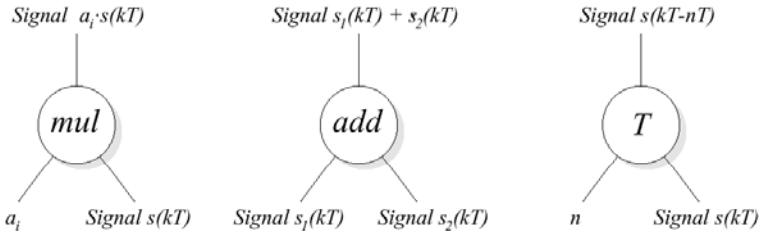
Anwendungsbeispiel: Digitale Filter



- Ziel: Suche nach einer funktionalen Beschreibung des Filters für geg. Impulsantwort

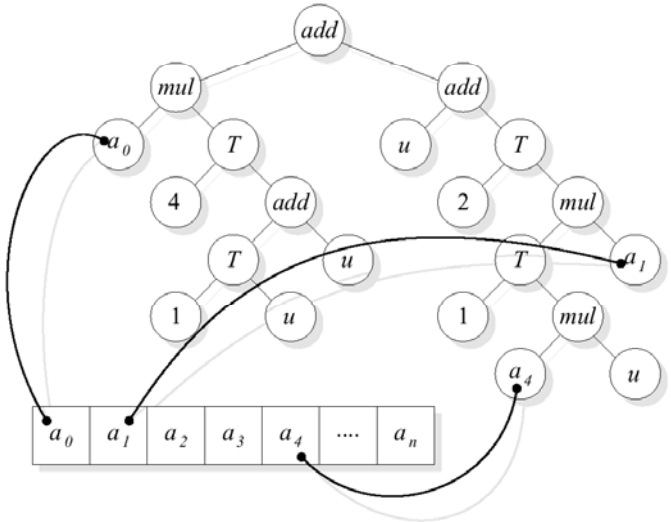
Anwendungsbeispiel: Digitale Filter

- Lösung durch hybride Struktur: Genetische Programmierung (Baumstrukturen) und Evolutionsstrategien (reelle Vektoren)
- Definition der Funktionsknoten: {MUL, ADD, T}



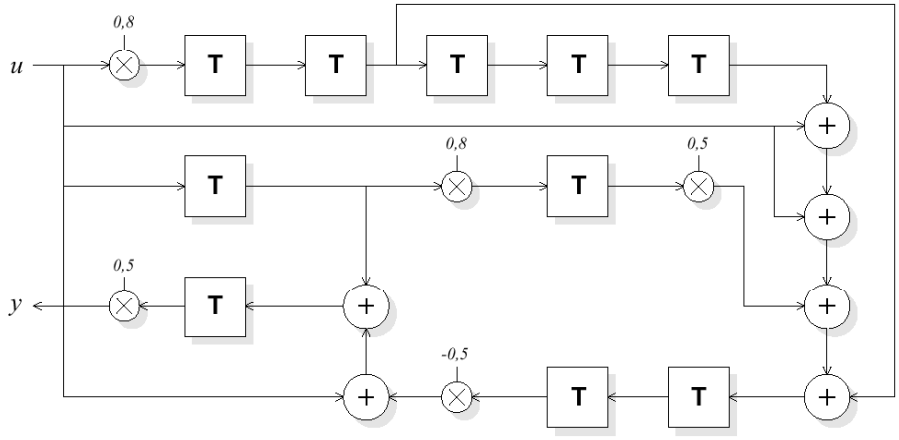
Anwendungsbeispiel: Digitale Filter

- Struktur der hybriden Lösungsrepräsentation:



Anwendungsbeispiel: Digitale Filter

– Beispiel einer gefundenen Lösung:



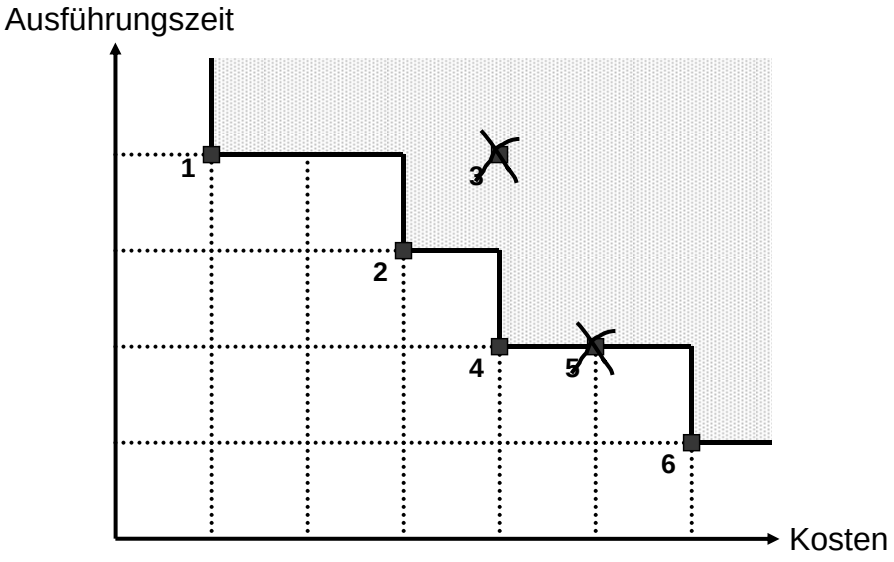
Übersicht

- Grundlagen Evolutionärer Algorithmen
- Strömungsrichtungen
- Hybride evolutionäre Algorithmen
- Systempartitionierung und -exploration

Entwurfsraumexploration

- Entwurfsraum
  - verschiedene Implementierungen einer Spezifikation
- Entwurfspunkt
  - eine Implementierung
- Optimierung
  - Bestimmung der “besten” Implementierungen, typisch: Mehrzieloptimierung
- Pareto-punkt
  - nichtdominierter Entwurfspunkt

Entwurfsraumexploration



Evolutionäre Ansätze

- Aggregationsfunktion
- Wechselnde Ziele: VEGA (Vector Evaluated Genetic Algorithm) [Schaffer84];
  - Selektion in n Teilschritte zerlegt; in jedem Schritt wird Selektion mit einem anderen Optimierungskriterium ausgeführt.
- Pareto-Ansätze:
  - Pareto-Ranking [Goldberg89]
  - Nischen-Techniken [Horn94]
  - Strength-Pareto-Verfahren [Zitzler, Thiele, Teich]

Evolutionäre Ansätze

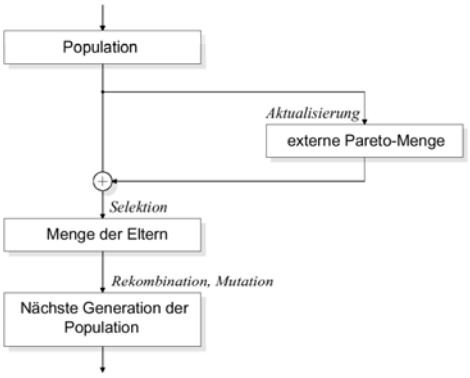
- Pareto-ranking:
  - Alle nichtdominierten Lösungen erhalten den Wert 1 und werden aus der Population gelöscht.
  - Nun werden die nichtdominierten Lösungen der Restpopulation ermittelt und mit dem Rang 2 versehen, usw.
  - ...
  - Nun kann ein Selektionsverfahren, z.B. rangbasierte Selektion angewendet werden.

Evolutionäre Ansätze

- Nischentechnik:
  - Verfahren kombiniert Wettkampfselektion mit dem Pareto-Ranking-Ansatz: Zwei Individuen und eine Vergleichsmenge von t_dom Individuen werden aus der Population gezogen.
  - Wenn ein Individuum von einem der Vergleichsmenge dominiert wird und das andere nicht, geht Letzteres als Sieger hervor.
  - Falls beide nichtdom. oder beide dom. werden, wird Sharing eingesetzt: Berechne, wieviele Individuen in der Nähe (Nischenradius) liegen.
  - Wähle das Individuum mit kleinerem Wert.

Das Strength-Pareto-Verfahren

- Pareto-optimale Individuen werden in einer externen Population gespeichert (Elitismus)
- Aktualisierung der externen Pareto-Menge
- Hohe Lösungsvielfalt in Pareto-Menge durch Clustering



Das Strength-Pareto-Verfahren

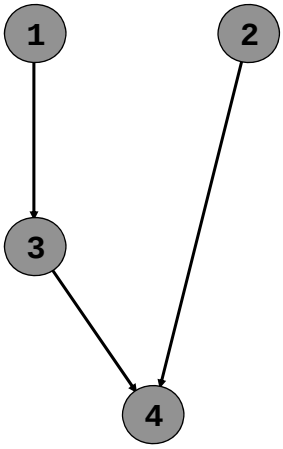
- Fitness-Bestimmung:
  - Individuen J der externen Pareto-Menge: Stärke (= Fitness) [0,1); Fitness von J proportional zur Anzahl der dominierten Individuen I der aktuellen Population.
  - Individuen I der aktuellen Population: Fitness gleich Summe der Stärken der Individuen der ext. Pareto-Menge, die I dominieren.
- Ziele:
  - Präferierung von Individuen nahe der Pareto-Front
  - „Volle“ Regionen werden bestraft durch große Stärken

Das Strength-Pareto-Verfahren

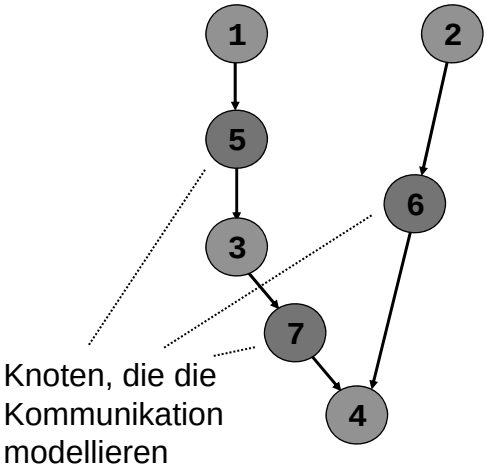
- Clustering:
  - Initialisierung: Jedes Individuum der ext. Pareto-Menge wird ein Cluster zugeordnet.
  - Zusammenfassen der Cluster: Diejenigen beiden Cluster mit geringstem mittleren Abstand (objective space) werden zu einem neuen Cluster vereinigt. Wiederhole solange, bis die Anzahl der Cluster klein genug ist.
  - Extrahieren der neuen Pareto-Menge: Jeder Cluster enthält eine Menge von Individuen. Behalte den Punkt in der Pareto-Menge, der den *Centroid* darstellt.

Anwendung: Systemsynthese

DFG



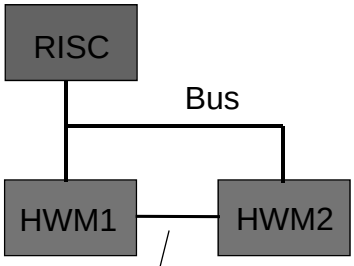
Problemgraph



Knoten, die die Kommunikation modellieren

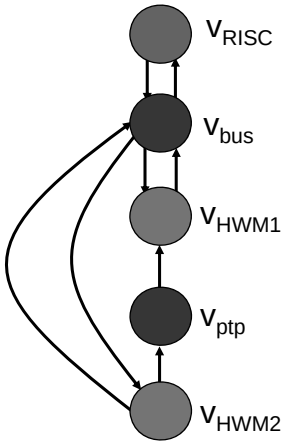
Anwendung: Systemsynthese

Architektur

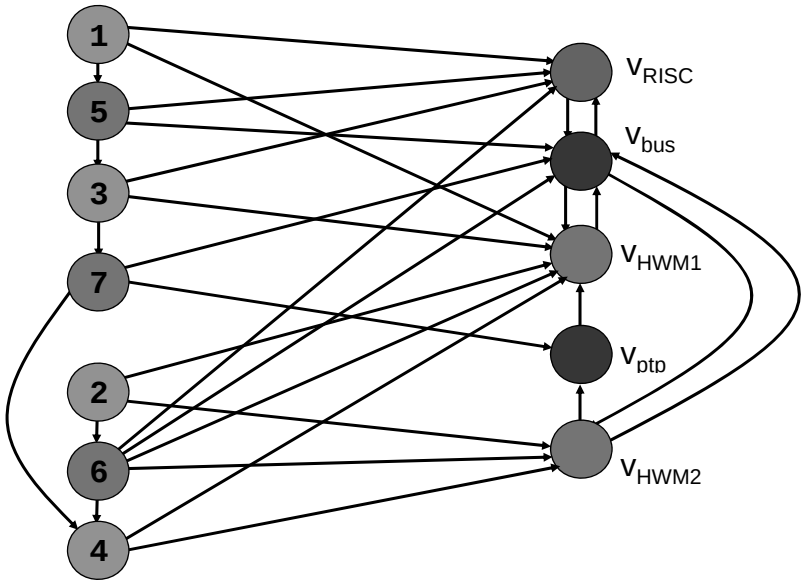


Punkt zu Punkt-Verbindung

Architekturgraph



Spezifikationsgraph



Allokation und Bindung

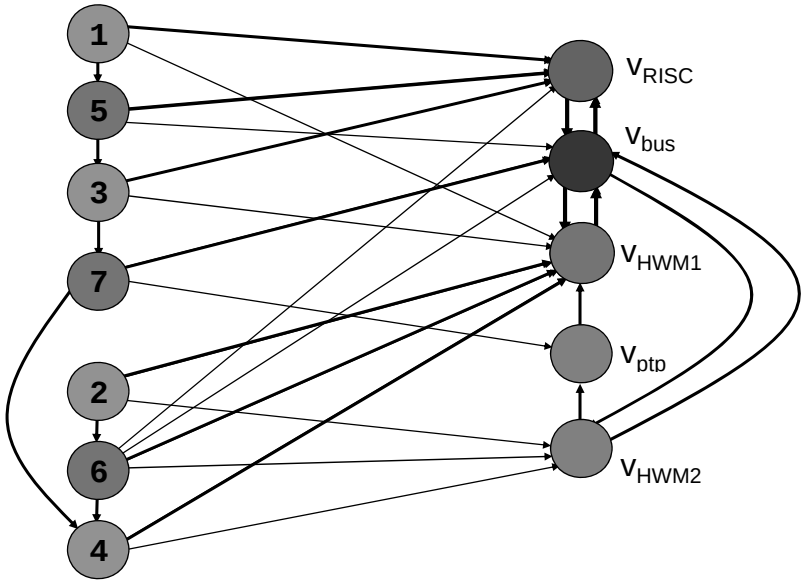
- Definition Allokation α: Teilmenge aller aktivierten Kanten und Knoten in G_P und G_A
- Definition Bindung β: Teilmenge aller aktivierten Abbildungskanten, insbesondere
  $$\beta = \{e \in E_M : \alpha(e) = 1\}$$
- Definition gültige Bindung:
  - $\forall e \in \beta : \alpha(pred(e)) = \alpha(succ(e)) = 1$
  - $\forall v \in nodes(G_P)$: $| \{e \in \beta \mid v = pred(e)\} | = 1$
  - $\forall e \in edges(G_P)$: entweder $(pred(e),x),(succ(e),x) \in \beta$ oder
    $(pred(e),x),(succ(e),y) \in \beta$ und $(x,y) \in \alpha$

Ergebnisse

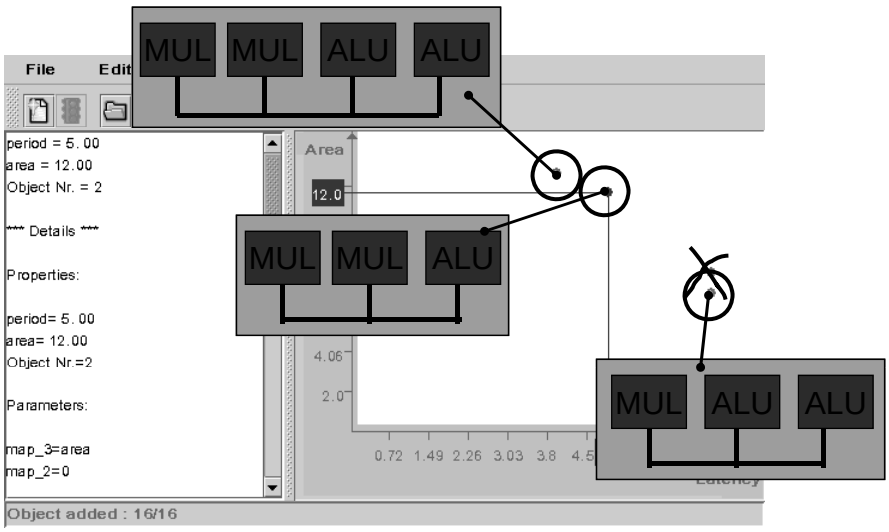
- Theorem [Teich 97, BTT98]:
  Die Bestimmung einer gültigen Bindung ist NP-vollständig.

- Ansatz unter Einsatz Evolutionärer Algorithmen: [BTT98]

Allokation, Bindung



Bsp.:Exploration Systemebene

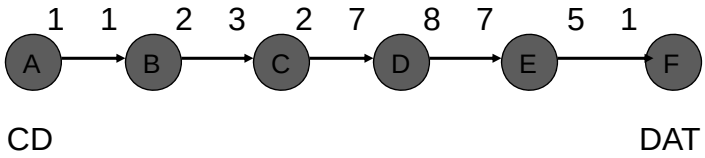


Anwendung: Softwaresynthese

- Spezifikation: Prozessgraph
- Target: Uniprozessor (DSP)
- Aufgaben:
  - Allokation von Programm- und Datenspeicher
  - Scheduling von Aktoren
  - Bindung an Register und Speicher
  - Auswahl des Implementierungsstils (z.B. Unterprogramm oder Inlining)

Anwendung: Softwaresynthese

- Beispiel: CD -> DAT Wandlung

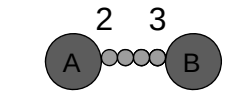


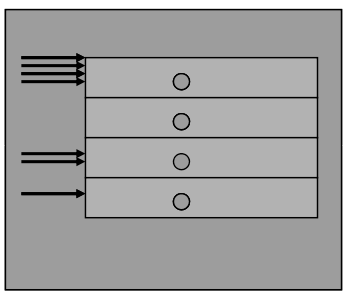
CD DAT

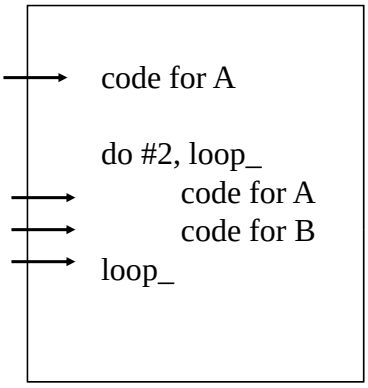
- (Schleifen-) Schedule:
  (∞(7(7(3AB)(2C))(4D))(32E(5F)))

Anwendung: Softwaresynthese

- Beispiel: Schedule (∞A(2AB))







Bsp.:Exploration Softwaresynthese

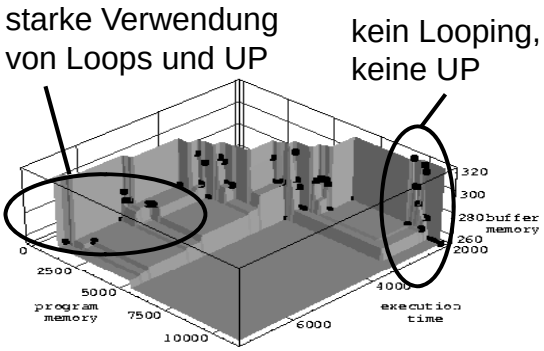
- Beispiel: CDtoDAT Benchmark
- Entwurfsraum:
  3D (Programmspeicher, Datenspeicher, Ausführungszeit)
- Parameterraum:
  - alle gültigen Loopschedules
  - UP vs. Inlining flag pro Aktor
- Optimierungsalgorithmus: Evolutionärer Algorithmus [Codes99,GECCO99,JVLSI00]

Bsp.:Exploration Softwaresynthese

- Programmspeicher (overhead):
  $P(S) = \Sigma(app(N_i,S)\ w(N_i)\ flag(N_i))$
  $+ (w(N_i) + app(N_i,S)\ P_S\ (1-flag(N_i))$
  $+ P_L(S)$
- Pufferspeicher (overhead):
  $D(S) = \Sigma\ max\_tokens(arc)$
- Ausführungszeit (overhead):
  $T(S) = (\Sigma(1-flag(N_i))\ L(N_i)\ q(N_i))$
  $+ IO(S) + O(S)$

Bsp.:Exploration Softwaresynthese

- Beispiel: CDtoDAT benchmark



Motorola DSP 56k

Zusammenfassung

- Stärken Evolutionärer Algorithmen:
  - breite Anwendbarkeit und Anpassbarkeit
  - tolerant gegenüber vorübergehender Verschlechterung des Zielfunktionswertes
  - keine restriktiven Annahmen über Zielfunktion, z.B. Stetigkeit, Differenzierbarkeit, etc.
  - gute Parallelisierbarkeit
- Schwächen Evolutionärer Algorithmen:
  - heuristischer Charakter
  - hoher Rechen- und Speicheraufwand
  - viele Freiheitsgrade