Algorithmen für Chip-Entwurfswerkzeuge

Timing-Analysen und Heuristiken

TECHNISCHE UNIVERSITÄT DARMSTADT

Vorlesung
WS 2013/2014

Florian Stock, Andreas Koch

Eingebette Systeme und Anwendungen
Technische Universität Darmstadt

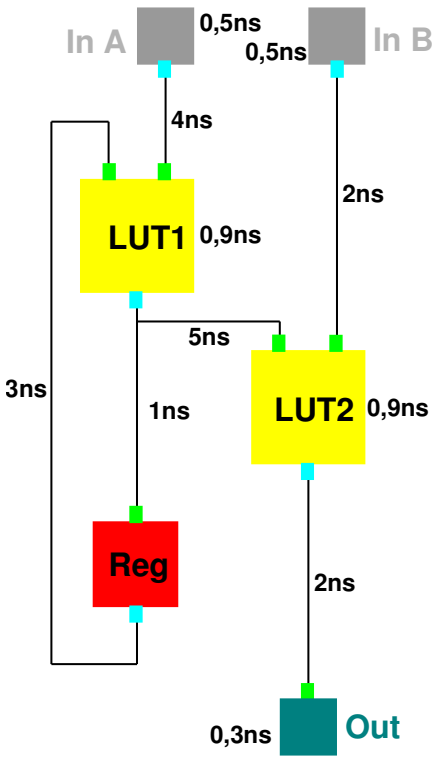
Embedded Systems & Applications

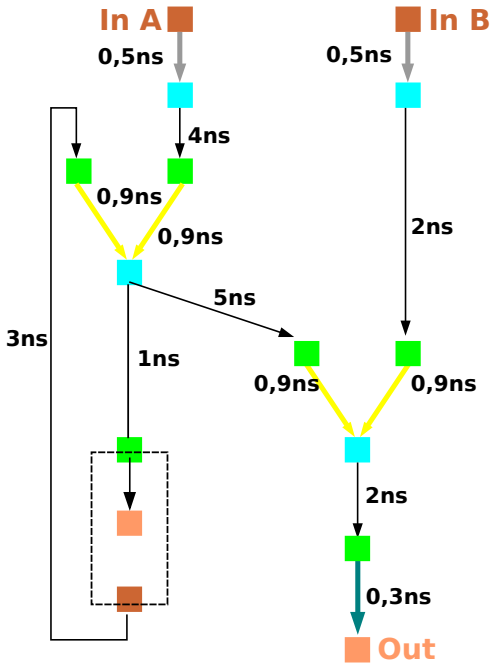
Timing-Analyse
Grundlagen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Wozu?
  - Analysiere fertige Layouts
  - Analysiere einzelne Verbindungen *während* Layouterzeugung
    - Erkenne kritische Verbindungen
    - Behandle diese mit Vorrang
- Worauf?
  - Schaltungselement
    - Gatter, Werttabellen (LUT), Register, I/O-Blöcke, ...
    - Bleiben konstant, exakte Verzögerungen bekannt
  - Netze
    - Nur nach Layouterzeugung bekannt
    - Vorher schätzen

Modellierung





- Auf *4-partitem* Graph
  - Externe Eingänge
  - Externe Ausgänge
  - Eingangs-Ports
  - Ausgangs-Ports

Berechnung Ankunftszeit

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ankunftszeit (arrival) an Knoten v
  $T_a(v) = \max_{(u,v)\in E}(T_a(u) + w(u, v))$
- Idee: BFS oder DAG LP
  - Beginne mit $T_a(v) = 0$ bei folgenden Knoten:
    Externer Eingänge, Registerausgänge
  - Bearbeite Knoten bei denen alle Vorgänger bearbeitet sind
  - Späteste Gesamtankunftszeit D_{max} = Taktperiode
    (an Externen Ausgängen, Registereingängen)

In Beispielschaltung

D_{max} = 13.6 ns

Spätestmögliche Ankunftszeit

TECHNISCHE UNIVERSITÄT DARMSTADT

- Wie unwichtig sind unkritische Netze?
  - Idee: Analog zu verschiebbaren Elementen bei Kompaktierung
  - Hier auf Zeitintervalle anwenden

Beantwortet die Frage

Wieviel langsamer kann ein Netz werden ohne das die gesamte Schaltung leidet?

- Berechung
  - Mittels spätestmöglicher Ankunftszeit
  - Required Time $T_r(v)$ an Knoten v
  - Spätestmöglicher Ankunftszeitpunkt von Signalen
    $\Rightarrow$ Sonst Verlangsamung der gesamten Schaltung
  - Analog Kompaktierungsbeispiel:
    Rechtestmögliche Position ohne Breitenvergrößerung

Berechnung
Spätestmögl. Ankunftszeit und Slack

TECHNISCHE UNIVERSITÄT DARMSTADT

- Beginne mit $T_r(u) = D_{max}$ bei folgenden Knoten:
  Externen Ausgänge, Registereingänge
- Nun BFS/LP Rückwärts
- Rückwärts $\Rightarrow$ Graph mit umgedrehten Kanten
- Bearbeite Knoten
  - Nur mit komplett bearbeiteten Vorgängern
  - $T_r(u) = \min_{(u,v)\in E}(T_r(v) - w(u,v))$
- *Slack (Schlupf)* einer Verbindung von u nach v
  $slack(u,v) = T_r(v) - T_u(u) - w(u,v)$
- Auf kritischem Pfad: $slack = 0$

Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

```
Node: 4  INPAD_SOURCE Block #2 (s27_in_3_)
T_arr: 0  T_req: -3.88578e-16  Tdel: 5e-10

Node: 5  INPAD_OPIN Block #2 (s27_in_3_)
Pin: 0
T_arr: 5e-10  T_req: 5e-10  Tdel: 5e-09
Net to next node: #2 (s27_in_3_).  Pins on net: 5.

Node: 12  CLB_IPIN Block #6 (s27_out)
Pin: 0
T_arr: 5.5e-09  T_req: 5.5e-09  Tdel: 0

Node: 17  SUBBLK_IPIN Block #6 (s27_out)
Pin: 0 Subblock #0
T_arr: 5.5e-09  T_req: 5.5e-09  Tdel: 9e-10

Node: 21  SUBBLK_OPIN Block #6 (s27_out)
Pin: 4 Subblock #0
T_arr: 6.4e-09  T_req: 6.4e-09  Tdel: 0

Node: 16  CLB_OPIN Block #6 (s27_out)
Pin: 4
T_arr: 6.4e-09  T_req: 6.4e-09  Tdel: 1e-09
Net to next node: #5 (s27_out).  Pins on net: 2.

Node: 10  OUTPAD_IPIN Block #5 (out:s27_out)
Pin: 0
T_arr: 7.4e-09  T_req: 7.4e-09  Tdel: 3e-10

Node: 11  OUTPAD_SINK Block #5 (out:s27_out)
T_arr: 7.7e-09  T_req: 7.7e-09

Tnodes on crit. path: 8  Non-global nets on crit. path: 2.
Global nets on crit. path: 0.
Total logic delay: 1.7e-09 (s)  Total net delay: 6e-09 (s)
```

Beispielanwendung
Unit-Size Placement Problem (UPP)

TECHNISCHE UNIVERSITÄT DARMSTADT

Eingabe

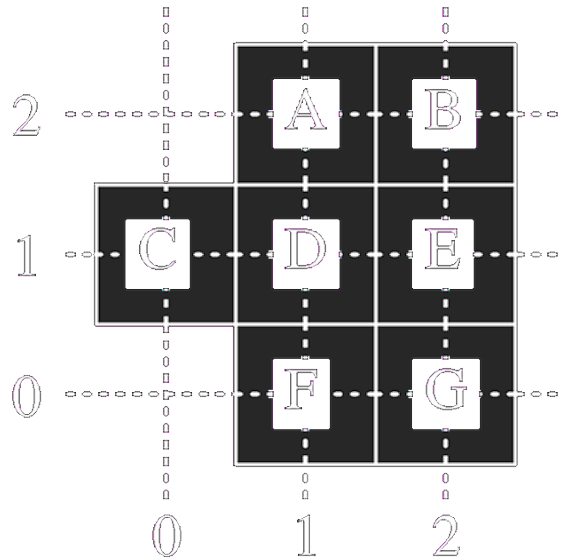
- 1×1 Zellen
- Netzliste

Platziere Zellen

- Auf 1×1 Raster
- Überlappungsfrei

Minimiere Fläche

- Platz für Verdrahtung



n_1: A, B, F, G

n_2: B, E

n_3: D, E

n_4: A, C, D

n_5: C, D, F

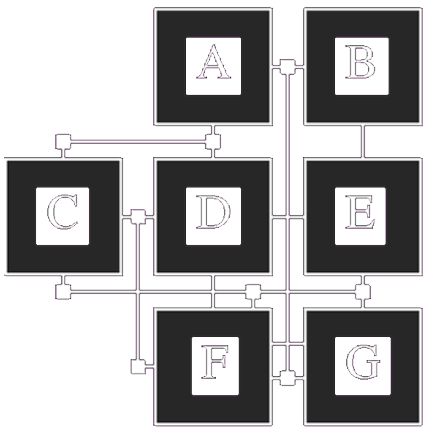
n_6: C, E, F, G

n_7: D, F

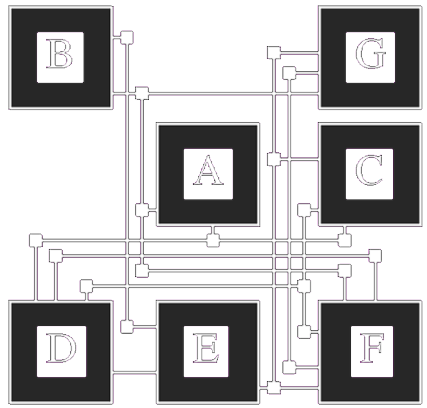
n_8: F, G

UPP

TECHNISCHE UNIVERSITÄT DARMSTADT



Gute Platzierung mit Verdrahtung



Schlechte Platzierung
Mehr Verdrahtungsspuren

Problem: Bestimmung der Qualität

- Komplette Verdrahtung dauert zu lange
- Abschätzen

Art der Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Viele Probleme im Bereich VLSI CAD sind
  - NP-vollständig
  - NP-hart (mindestens so aufwendig wie NP-vollständig)
- Exakt lösbar nur für kleine Problemgrößen
- Falls suboptimale Lösungen aktzeptabel:
  - Näherungsverfahren, Approximationen
    - Garantieren eine bestimmte Lösungsqualität
    - Nicht allgemein formulierbar
  - Heuristiken
    - Schwankende Lösungsqualität

Darstellung einer *Lösung*

TECHNISCHE UNIVERSITÄT DARMSTADT

- Problemspezifisch
- Algorithmenspezifisch
- Grundsätzlich unterscheidbar
  - Vollständige Lösung
    - Alle Unbekannten haben gültige Werte
    - Algorithmus könnte beliebig beendet werden
  - Unvollständige Lösung
    - Einige/Alle Unbekannte sind noch unbestimmt oder ungültig
    - Algorithmus muß weiterrechnen

Lösungsdefintion

TECHNISCHE UNIVERSITÄT DARMSTADT

- Probleminstanz $I = (F, c)$
  - Lösungsraum F
  - Kostenfuntion $c : F \mapsto \mathbf{R}$
- Lösung $\vec{f} \in F : \vec{f} = [f_1, ... f_n]^T$
  - Explizite Einschränkungen: Wertebereiche der f_i
  - Implitizite Einschränkunge: Abhängigkeiten
- Teillösung $\vec{f}$
  - Einige f_i undefiniert
  - Spannt Unterraum von F auf

Nachbarsuche
Idee

TECHNISCHE UNIVERSITÄT DARMSTADT

- Starte mit einer vollständigen Lösung
- Bestimme *Nachbarn* der Lösung
  - Andere Lösungen *nahe* an existierender
  - Definition von *Nähe* ist problemspezifisch
- Wähle *besseren* Nachbarn aus
- Wiederhole

Nachbarsuche
Formal

TECHNISCHE UNIVERSITÄT DARMSTADT

- Problem $I = (F, c)$
- Lösung $\vec{f} \in F$
- Nachbarschaft $N : F \mapsto 2^F$
  (2^F Potenzmenge = Menge aller Teilmengen von F)
- Nachbar $\vec{g} \in N(\vec{f})$

Beispiel UPP

- $\vec{g}$ ist Nachbar von $\vec{f}$:$\Leftrightarrow$ $\vec{g}$ ergibt sich durch Vertauschung von 2 Zellen von $\vec{f}$.
- n Zellen, $n-1$ Tauschpartner $\Rightarrow \left|N(\vec{f})\right| = \frac{n(n-1)}{2}$
- Komplexere *Züge* möglich
  z.B. tausche 3 Zellen, tausche Regionen, ...

Nachbarsuche
Kandidatenwahl

TECHNISCHE UNIVERSITÄT DARMSTADT

- Welches $\vec{g} \in N(\vec{f})$ wählen?
- Ziel: Kostenreduzierung bezüglich c
- Also wähle $\vec{g}$ mit $c(\vec{g}) < c(\vec{f})$
- Ende mit $\vec{f}$ falls $\forall \vec{g} \in N(\vec{f}) : c(\vec{g}) \geq c(\vec{f})$

Nachbarsuche
Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

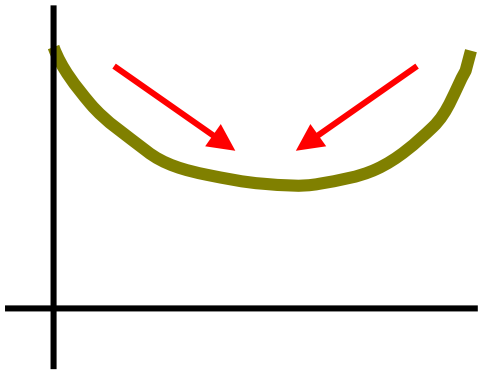
```
Local_search() : begin
    Feasible_solution f ;
    Set<Feasible_solution> G ;
    f := Initial_solution() ;
    repeat
        G := {g|g ∈ N(f) ∧ c(g) < c(f)} ;
        if G ≠ ∅ then
            f := G.pickany()
    until G = ∅;
    report(f) ;
```

- Initialisierung: `Initial_solution`? Zufällig, triviale Lösung
- Strategien: `pickany`? Erste Verbesserung, Steilster Abstieg

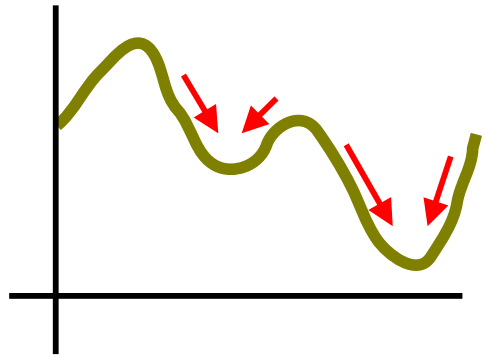
Nachbarsuche
Probleme

TECHNISCHE UNIVERSITÄT DARMSTADT

- *Form* der Kostenfunktion:



Gut geeignet



Schlecht geeignet

- Steckenbleiben in lokalen Minima
- Mögliche Lösungen:
  - Mehrere Läufe mit anderen Startlösungen
  - Größere Nachbarschaft
  - Adaptiere Größe der Nachbarschaft

Simulated Annealing
Idee

TECHNISCHE UNIVERSITÄT DARMSTADT

- Aktzeptiere verschlechternde Züge
  Aber bessere Strategie als reiner Zufall
- Simulated Annealing
  - Simuliertes Erstarren von Metallen
  - Inspirriert von physikalischen Erstarrungsprozessen
    - Schnelles Erstarren (*Schockfrosten*)
      Hohe innere Spannung = hohe innere Energie
    - Langsames Abkühlen Niedrige innere Spannung = niedrige innere Energie

Simulated Annealing
Physikalischer Hintergrund

TECHNISCHE UNIVERSITÄT DARMSTADT

- Hohe Anfangstemperatur (flüssiges Material)
- Moleküle können sich frei anordnen
- Langsames Abkühlen
- Bewegungsfreiheit wird schrittweise weiter eingeschränkt
- Moleküle ordnen sich in Konfiguration niedrigster Energie an
- Am besten bei sehr langsamer Abkühlung

Simulated Annealing für die Optimierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Energie entspricht Kostenfunktion
- Bewegung der Moleküle entspricht Zügen
- Temperatur entspricht Kontrollparameter T
  - Wie frei dürfen sich Moleküle bewegen?
    = Welche Züge sind noch akzeptabel?
  - Niedrigere Energie/Kosten: Immer akzeptiert
    $c(\vec{g}) \leq c(\vec{f})$
  - Höhere Energie/Kosten:
    Akzeptiert mit Wahrscheinlichkeit $e^{\frac{-\Delta c}{T}}$ mit $\Delta c = c(\vec{g}) - c(\vec{f})$

Simulated Annealing
Temperatur

TECHNISCHE UNIVERSITÄT DARMSTADT

$$e^{\frac{-\Delta c}{T}} = \frac{1}{e^{\frac{\Delta c}{T}}}$$

- Hohe Temperaturen
  Akzeptiere fast alle schlechten Züge
- Niedrige Temperaturen
  Akzeptiere fast keine schlechten Züge mehr
- Physik: Boltzmann-Verteilung
  Statistische Mechanik

Simulated Annealing Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

```
Simulated_annealing(): begin
    Feasible_solution f, g, bsf ;
    float T ;
    T := Initial_temperature() ;
    f := Initial_solution() ;
    bsf := f ;
    repeat
        repeat
            g := N(f).pickany() ;
            if Accept(f, g) then  f := g ;
        until Thermal_equilibrium(T);
        T := New_temperature(T) ;
    until Stop();
    report(bsf) ;
```

```
int Accept(Feasible_solution f, g):
begin
    float DeltaC ;
    DeltaC := c(g) - c(f) ;
    if DeltaC ≤ 0 then
        if c(g) < c(bsf) then  bsf := g ;
        return (1);
    else
        return (exp(-DeltaC/T) >
        random(1))
```

Simulated Annealing
Zusätzliche Funktionen

TECHNISCHE UNIVERSITÄT DARMSTADT

Initial_temperature()

Bestimmt ausreichend hohe Starttemperatur

Initial_solution()

- Bestimmt Startlösung
- Zufällige, aber gültige Lösung OK!

Thermal_equilibrium()

Gleichgewicht auf einer Temperaturstufe

New_temperature()

Bestimmt nächsten Temperaturschritt

Stop()

Abbruchkriterium

BSF: *Best so far*

- Beste bisherige Lösung
- Letzte Lösung ist nicht immer die beste!

Simulated Annealing
TimberWolf (1985)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Standard Cell-Placer
- Start mit $T = 4.000.000$
- Stop bei $T < 0.1$
- Equilibrium abhängig von Problemgröße
  - 100 Züge pro Zelle bei 200 Zellen
  - 700 Züge pro Zelle bei 3000 Zellen
- Abkühlen
  - Anfangs mit $T_n = 0.8T$
  - Im Mittelbereich mit $T_n = 0.95T$
  - Gegen Ende mit $T_n = 0.8T$

  $\Rightarrow$ Cooling Schedule

Simulated Annealing

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bei geeigneter Cooling Schedule
  - SA findet immer die optimale Lösung
  - Praktisch aber nicht relevant (zu langsam)
- Viele Variationsmöglichkeiten
  - stop() abhängig von accept()
  - Adaptive Cooling Schedules
- Bibliotheken: ASA, EBSA
- SA ist allgemein verwendbar
- Aber: Spezialisierte Lösungen sind besser

Tabu-Suche

TECHNISCHE UNIVERSITÄT DARMSTADT

- Simulated Annealing
  - Verschlechternde Züge zu Beginn akzeptiert
- Tabu-Suche (TS)
  - Verschlechternde Züge werden immer akzeptiert
  - Gehe immer zu $\vec{g} \in N(\vec{f})$ mit $c(\vec{g}) = \min_{\vec{h} \in N(\vec{f})} c(\vec{h})$
  - Auch wenn $c(\vec{g}) > c(\vec{f})$!
  - Problem: Zyklen
    - Ständige Wiederholung der letzten Züge
- Lösung: Verbiete letzten k Lösungen
  - Lösungen sind als *tabu* markiert
  - Vermeidet Zyklen bis zu der Länge k
  - Realsierung: FIFO mit Lösungen der Länge k

Tabu-Suche
Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

```
Tabu_search(): begin
    Feasible_solution f, g, bsf ;
    Set<Feasible_solution> G ;
    FIFO<Feasible_solution,k> Q ;
    f := Initial_solution(); bsf := f; Q := ∅ ;
    repeat
        G := {s|s ∈ N(f) ∧ s ∉ Q} ;
        if G ≠ ∅ then
            g := G.findmin(c) ;
            Q.shiftin(g) ;
            f := g ;
            if c(f) < c(bsf) then bsf := f;
    until (G ≠ ∅) or Stop();
    report(bsf);
```

Tabu-Suche

TECHNISCHE UNIVERSITÄT DARMSTADT

- stop()
  Keine Verbesserung in den letzten k Zügen

UPP-Beispiel

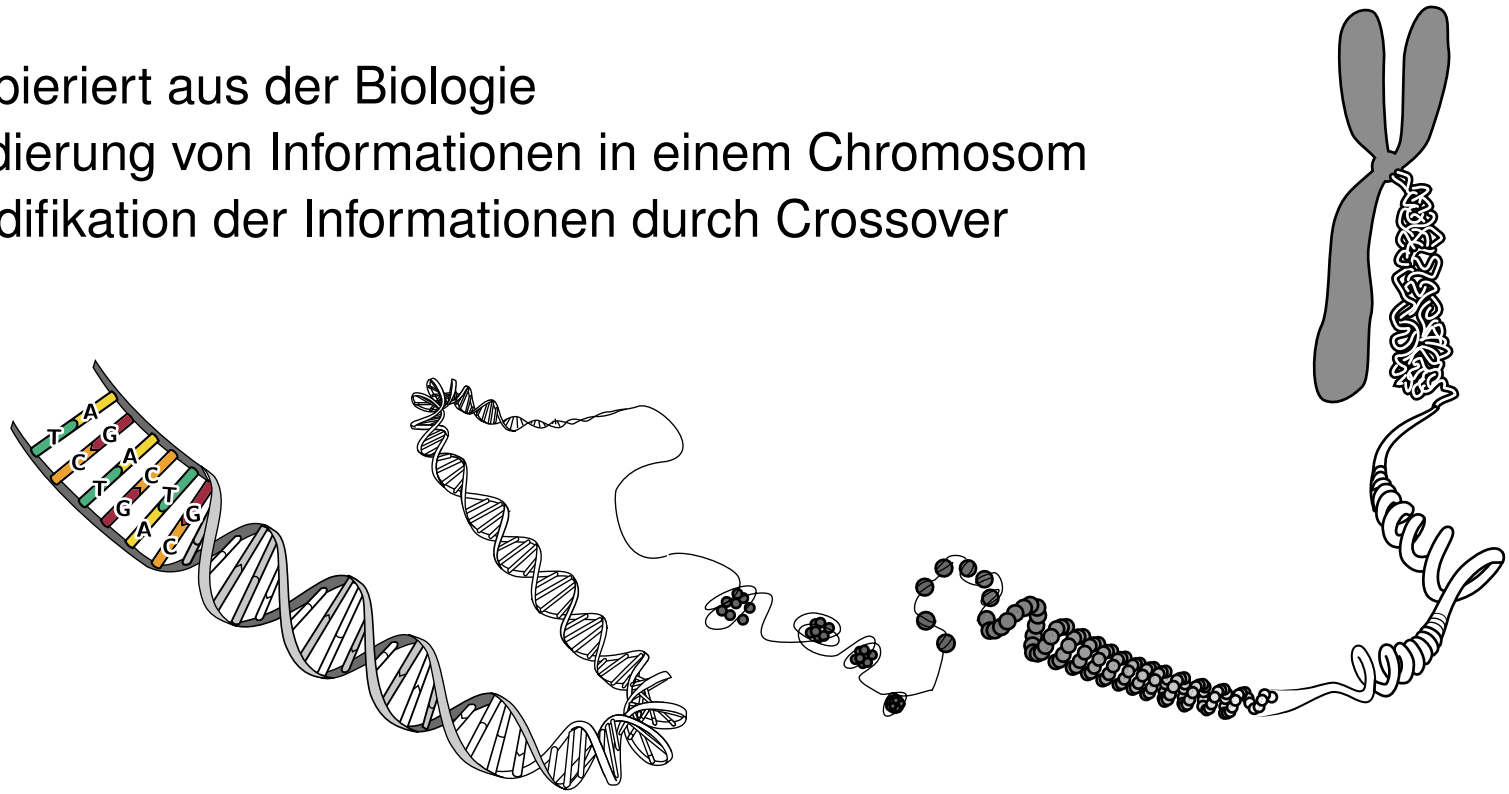
- 10.000 Zellen Lösung beschreibt 10.000 Koordinatenpaare
- Sehr große Tabu-Liste
- Abhilfe: Setze nur einzelne Züge Tabu
- Aber: Einschränkung des Lösungsraumes

- Viele Variationsmöglichkeiten
  - Kein theoretischer Hintergrund Erreichen des Optimums?
  - Wie stop() oder k wählen?

Genetische Algorithmen
Idee

TECHNISCHE UNIVERSITÄT DARMSTADT

- Inspieriert aus der Biologie
- Kodierung von Informationen in einem Chromosom
- Modifikation der Informationen durch Crossover



Basiert auf einem Bild der NIH

Genetische Algorithmen
Idee

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bisher: Algorithmen arbeiten auf *einer* vollständigen Lösung
- Nun: Gleichzeitig auf *mehreren* Lösungen
  - Menge P von Lösungen: Population
  - Iterationszähler k: Generation
  - Ersetze $P^{(k)}$ durch $P^{(k+1)}$ während Optimierung
- Bestimmung von $\vec{f}^{(k+1)} \in P^{(k+1)}$ mittels
  - Crossover von zwei Eltern $\vec{f}^{(k)}, \vec{g}^{(k)} \in P^{(k)}$
    Vererbung von Eigenschaften von $\vec{f}^{(k)}$ und $\vec{g}^{(k)}$
  - Ggf. Mutation von $\vec{f}^{(k+1)}$

Genetische Algorithmen
Darstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Kodierung für Chromosom
  Bitfolge für Lösungsvektor

UPP Kodierung

- 100 Zellen, 10 $\times$ 10 Raster
  - 4 Bit pro Koordinate
  - 8 Bit pro Koordinatenpaar
  - 100 $\times$ 8 Bit = 800 Bit lange Bitfolge als Chromosom
  - L Länge des Chromosoms in Bit

- Wichtige Unterscheidung zwischen
  Lösung Biologie: Phänotyp
  Kodierung der Lösung Biologie: Genotyp
- Hier im Beispiel äquivalent benutzt

Genetische Algorithmen
Vererbung mit dem Crossover-Operator

TECHNISCHE UNIVERSITÄT DARMSTADT

- Kombiniere die Bitfolgen der Eltern
- Verschiedene Realisierungen möglich

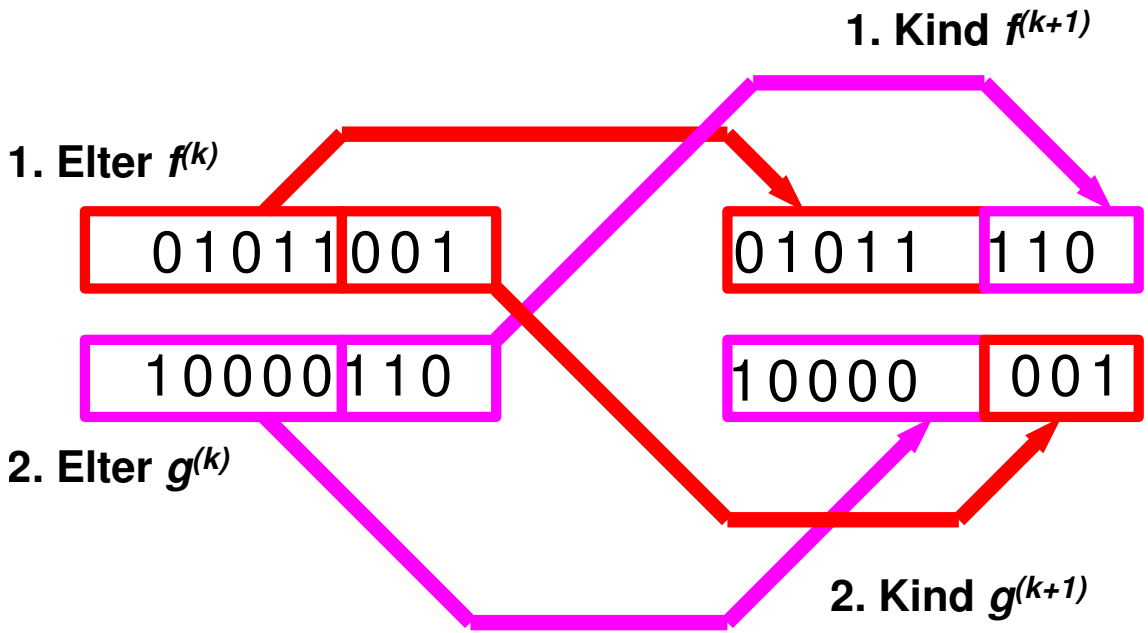
Beispiel

1. Wähle zufällig Crossover-Position $1 \leq r \leq L$
2. Kopiere Bits $1 \ldots (r-1)$ aus $\vec{f}^{(k)}$ nach $\vec{f}^{(k+1)}$
3. Kopiere Bits $r \ldots L$ aud $\vec{g}^{(k)}$ nach $\vec{f}^{(k+1)}$
4. Ggf. erzeuge 2. Kind $\vec{g}^{(k+1)}$ mit vertauschten Rollen

Genetische Algorithmen
Beispiel UPP

TECHNISCHE UNIVERSITÄT DARMSTADT

10×10 Raster, platziere einzelne Zelle



$r = 6$

Genetische Algorithmen
Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Crossover erzeugt ungültige Lösungen

$\Rightarrow$ Abhilfe: Mehr Struktur als einfache Bitfolge

Bei UPP-Beispiel

- Folgen von 4-bit Koordinaten
- Nun zwar intern konsistente Koordinaten
- Reicht im allgemeinen Fall aber nicht aus!

$\Rightarrow$ Problemspezifisches Crossover

Genetische Algorithmen
Optimierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bisher noch keine Optimierung
  Nur neue Lösungen erzeugt
- Bevorzuge gute Lösungen vor schlechten
  - Wähle *gute* Eltern aus: Niedrige Kosten
  - Kombiniere gute Eigenschaften in Nachwuchs
  - Aber: Auch Gegenteil möglich (Crossoverposition *r* zufällig)
    - Vererbung schlechter Eigenschaften
    - Idee: Schlechte Nachkommen sterben in nächster Generation

Genetische Algorithmen
Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

```
Genetic_Algorithm(int popsize) : begin
    Set<chromosome> pop, newpop ;
    Chromosome parent1, parent2, child ;
    pop := ∅ ;
    for i:=1 . . . popsize do  pop := pop ∪ {Chromosom einer zufälligen Lösung } ;
    repeat
        newpop := ∅ ;
        for i:=1 . . . popsize do
            parent1 := pop.select() ; parent2 := pop.select() ;
            child := crossover(parent1, parent2) ;
            newpop : = newpop ∪ { child };
        pop := newpop;
    until Stop();
    report(pop.findmin(c));
```

Genetische Algorithmen

TECHNISCHE UNIVERSITÄT DARMSTADT

- `Stop()`
  - Keine Verbesserung in den letzten m Iterationen
  - m problemspezifischer Parameter
- Mutation
  - Modelliert in Natur aufkommende Fehler beim Kopieren/durch Fremdeinwirkungen
  - Vermeidet Steckenbleiben in lokalen Minima
- Sehr viele Variationsmöglichkeiten
  - Komplexere Crossover (mehrer r,...)
  - Zusätzliche Funktionen: Mutation, Inversion
  - Mehere Generationen gleichzeitig
  - Elite-Selektion
  - Meta-Genetische Algorithmen

...

Genetische Algorithmen
Genie (1987)

TECHNISCHE UNIVERSITÄT DARMSTADT

Stop(): 10.000 Generationen ohne Verbesserung der BSF-Lösung

Initiale Population: Anzahl: Hunderte
25% Zufällig, 75% sequentiell angeordnet

.select(): Zufällig mit Gewichtung auf *fitness* ($< \varnothing$ nie)

Resultat: Ähnlich gut wie TimberWolf
(Tendenziell bessere Ergebnisse auf Kosten der Laufzeit)

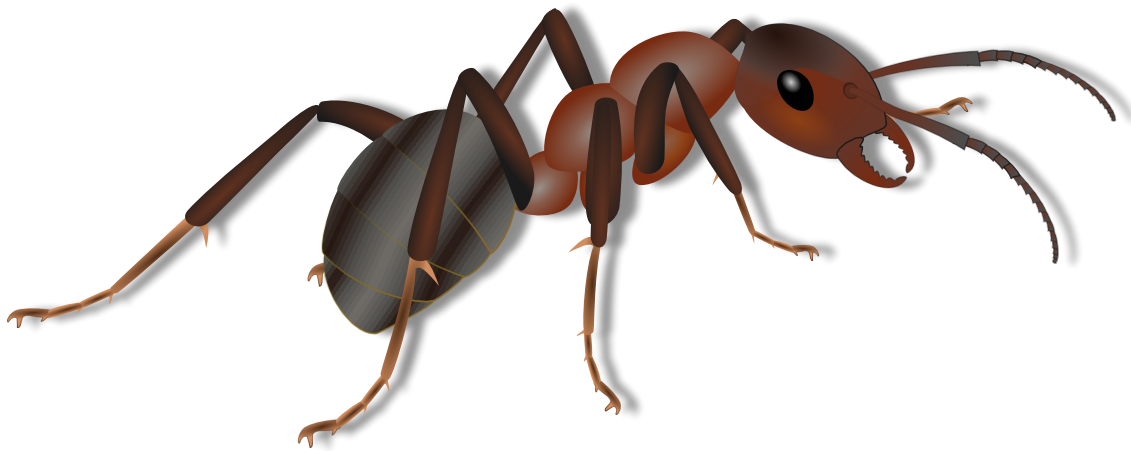
Ameisenalgorithmus
Idee

TECHNISCHE UNIVERSITÄT DARMSTADT

- Inspiriert von Ameisenpfaden
- Ameisen hinterlassen beim marschieren Pheromone
- Kurze Wege akkumulieren stärkere Pheromonkonzentration

$\Rightarrow$ werden entsprechend attraktiver für andere Ameisen

- Agentenbasierter Ansatz: Finden von besten Wegen im Graph



Ameisenalgorithmus
Anwendung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Oft benutzt bei *TSP* (Travelling-Salesman-Problem) oder QAP

QAP (Quadratic-Assignment-Problem)

Gegeben:

$f \in F$	Einrichtungen
$l \in L$	Orte
$d(l_1, l_2)$	Distanz zwischen zwei Orten ($d : L \times L \mapsto R$)
$w(f_1, f_2)$	Fluß zwischen zwei Einrichtungen ($w : F \times F \mapsto R$)

Gesucht: Zuordnung $a : F \mapsto L$ bei der $\sum_{f_1, f_2 \in F} w(f_1, f_2) \cdot d(a(f_1), a(f_2))$ minimal wird

- Lösung wird iterativ in jedem *Agenten* (Ameise) aufgebaut
- Jede Ameise läuft zufälligen Weg
  Wahrscheinlichkeit bei Weg mit vielen Pheromonen höher
- Nach Konstruktion des Wege, entsprechend der Kosten Pheromongehalt der Wege aller Ameisen aktualisieren

Ameisenalgorithmus
UPP-Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Zellen $\mapsto$ Einrichtungen
- Orte $\mapsto$ Positionen im Raster
- Fluß $\mapsto$ Anzahl der adjazenten Zellen
- Distanz $\mapsto$ Distanz

Ameisenalgorithmus
Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

```
Ant_colony_optimization:
begin
    Init_heuristics() ;
    repeat
        forall the Ants i do
            Ant i.generate_solution()
        Update_pheromons() ;
    until Stop();
```

```
Ant.generate_solution():
begin
    for NumberCells do
        Select_cell() ;
        Assign_cell_to_location()
    Improve_solution() ;
    if solution < bsf then
        bsf := solution
```

Stop(): Abbruchkriterium: Anzahl Iterationen abhängig von Problemgröße, keine Änderung der BSF, ...

Improve_solution(): Lokale Suche zum verbessern des Ergebnisses

Select_cell(): Zufällig, feste Reihenfolge (z.B. sortiert nach Summe der inzidenten Flüße)

Ameisenalgorithmus
Initialisierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Heuristische Information

$$\eta_{ij} = \frac{1}{f_i \cdot d_j}$$

f_i *Flußpotenial* der Zelle (Anzahl verbundener Zellen)
d_i *Distanzpotial* einer Position
(Summe der Distanzen zu allen anderen Positionen)

- Setzen sonstiger Parameter:
  - k Anzahl Ameisen
  - Q Wert der Pheromonerhöhung (abhängig von Zielfunktion)
  - $\alpha, \beta > 0$ Gewichtung der Pheromone (α) und heuristischen Information (β)
  - ρ Persistenz der Pheromone
  - τ_{ij} Pheromone, Initialisierung mit Wert > 0

Ameisenalgorithmus
Zuweisung Zelle $\mapsto$ Position

TECHNISCHE UNIVERSITÄT DARMSTADT

Wahrscheinlichkeit das Zelle i Position j zugewiesen wird (in Iteration t der Ameise k):

$$p_{ij}^k = \frac{\tau_{ij}(t)^\alpha \cdot \eta_{ij}^\beta}{\sum_{l \in N_i^k} \tau_{il}(t)^\alpha \cdot \eta_{il}^\beta}$$

- N_i^k ist die Nachbarschaft von Knoten i
- $\sum_{l \in N_i^k} p_{il}(t) = 1$
- τ_{ij} Pheromonwerte, η_{ij} heuristische Information

Ameisenalgorithm
Update

TECHNISCHE UNIVERSITÄT DARMSTADT

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

Mit

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/cost(solution) & \text{Zelle } i \text{ ist Position } j \text{ zugeordnet} \\ 0 & \text{sonst} \end{cases}$$

Q Menge der Pheromone einer Ameise

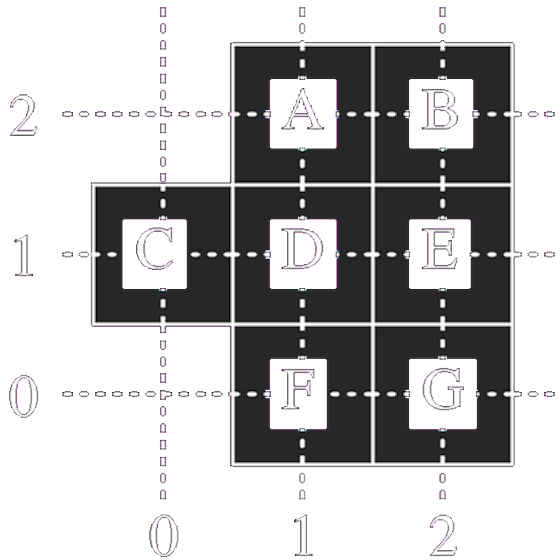
$0 < \rho < 1$ Persistenz der Pheromone ($(1 - \rho)$ Evaporation)
Modelliert verdampfen von Pheromonen (gegen lokale Minima)

Ameisenalgorithmus
Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Schlechte Konvergenz in der Nähe des Optimums, da kaum ein Unterschied im Pheremonlevel
- Schlechte Skalierung
- Qualität und Laufzeit vergleichbar mit Simulated Annealing
- Änlich zu SA: Theoretisch optimal
- Kann dynamisch auf Graphänderungen reagieren
- Viele Varianten: AS, ANTS, MAX-MIN, FANT, HAS-QAP, ... Hauptunterschied sind andere ...
  - heuristische Informationen
  - Pheromon Updates
  - Wahrscheinlichkeitsfunktion

Ameisenalgorithmus
Beispiel UPP



n_1: A, B, F, G

n_2: B, E

n_3: D, E

n_4: A, C, D

n_5: C, D, F

n_6: C, E, F, G

n_7: D, F

n_8: F, G

$$\vec{f} = \begin{pmatrix} 5 \\ 4 \\ 7 \\ 6 \\ 5 \\ 10 \\ 7 \end{pmatrix}, \vec{d} = \begin{pmatrix} 18 \\ 15 \\ 18 \\ 15 \\ 12 \\ 15 \\ 18 \\ 15 \\ 18 \end{pmatrix}$$

Zellen sortiert nach Flußpotenial: F, C, G, D, A, E, B

Ameisenalgorithmus
Beispiel UPP – Init

TECHNISCHE UNIVERSITÄT DARMSTADT

$$\eta = \begin{pmatrix} 0.0111 & 0.0139 & 0.0079 & 0.0093 & 0.0111 & 0.0056 & 0.0079 \\ 0.0133 & 0.0167 & 0.0095 & 0.0111 & 0.0133 & 0.0067 & 0.0095 \\ 0.0111 & 0.0139 & 0.0079 & 0.0093 & 0.0111 & 0.0056 & 0.0079 \\ 0.0133 & 0.0167 & 0.0095 & 0.0111 & 0.0133 & 0.0067 & 0.0095 \\ 0.0167 & 0.0208 & 0.0119 & 0.0139 & 0.0167 & 0.0083 & 0.0119 \\ 0.0133 & 0.0167 & 0.0095 & 0.0111 & 0.0133 & 0.0067 & 0.0095 \\ 0.0111 & 0.0139 & 0.0079 & 0.0093 & 0.0111 & 0.0056 & 0.0079 \\ 0.0133 & 0.0167 & 0.0095 & 0.0111 & 0.0133 & 0.0067 & 0.0095 \\ 0.0111 & 0.0139 & 0.0079 & 0.0093 & 0.0111 & 0.0056 & 0.0079 \end{pmatrix}$$

$$Q = 2, \alpha = \beta = 1, \rho = 0.8$$

Ameisenalgorithmus
Beispiel UPP – Init

$$\tau = \begin{pmatrix} 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \end{pmatrix}$$

$$Q = 2, \alpha = \beta = 1, \rho = 0.8$$

Ameisenalgorithmus
Beispiel UPP – Auswahl

TECHNISCHE UNIVERSITÄT DARMSTADT

$$p_{ij} = \frac{\tau_{ij} \cdot \eta_{ij}}{\sum_{l \in N_i^k} \tau_{il} \cdot \eta_{il}}$$

Platziere Zelle F

τ_{Fi}	0.0056	0.0067	0.0056	0.0067	0.0083	0.0067	0.0056	0.0067	0.0056
η_{Fi}	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
$\tau_{Fi} \cdot \eta_{Fi}$	0.00056	0.00067	0.00056	0.00067	0.00083	0.00067	0.00056	0.00067	0.00056

$\sum = 0.00575$

p_{Fi}	0.0973	0.1165	0.0973	0.1165	0.1443	0.1165	0.0973	0.1165	0.0973

Ameisenalgorithmus
Beispiel UPP – Auswahl

TECHNISCHE UNIVERSITÄT DARMSTADT

$$p_{ij} = \frac{\tau_{ij} \cdot \eta_{ij}}{\sum_{l \in N_i^k} \tau_{il} \cdot \eta_{il}}$$

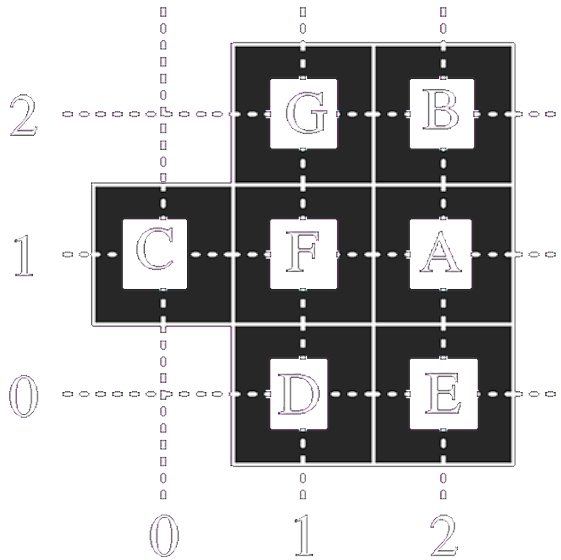
Platziere Zelle C

τ_{Fi}	0.0079	0.0095	0.0079	0.0095	0.0079	0.0095	0.0079	0.0095	0.0079
η_{Fi}	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
$\tau_{Fi} \cdot \eta_{Fi}$	0.00079	0.00095	0.00079	0.00095		0.00095	0.00079	0.00095	0.00079
$\sum$ = 0.00696									
p_{Fi}	0.1135	0.1365	0.1135	0.1365		0.1365	0.1135	0.1365	0.1135

Ameisenalgorithmus
Beispiel UPP

TECHNISCHE UNIVERSITÄT DARMSTADT

- Analog G, D, A, E, B



- Kosten (HPWL): 24
- Nächster Schritt: Pheromone aktualisieren

Ameisenalgorithmus
Beispiel UPP – Update

$$\tau = \begin{pmatrix} 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \end{pmatrix}$$

Ameisenalgorithmus
Beispiel UPP – Update

$$\tau = \begin{pmatrix} 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 \\ 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 \\ 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 \\ 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 \\ 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 \\ 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 \\ 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 \\ 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 \\ 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 \end{pmatrix}$$

$\rho = 0.8$ Evaporation

Ameisenalgorithmus
Beispiel UPP – Update

TECHNISCHE UNIVERSITÄT DARMSTADT

$$\tau = \begin{pmatrix} 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 \\ 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.163 \\ 0.08 & 0.163 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 \\ 0.08 & 0.08 & 0.0163 & 0.08 & 0.08 & 0.08 & 0.08 \\ 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.163 & 0.08 \\ 0.163 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 \\ 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 \\ 0.08 & 0.08 & 0.08 & 0.163 & 0.08 & 0.08 & 0.08 \\ 0.08 & 0.08 & 0.08 & 0.08 & 0.163 & 0.08 & 0.08 \end{pmatrix}$$

$\frac{2}{24}$ auf genommenen Pfad hinzufügen

Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Timinganalyse
- UPP
- Allgemeine Heuristiken
  - Nachbarsuche
  - Simulated Annealing
  - Tabu-Suche
  - Genetische Algorithmen
  - Ameisenalgorithmus