INF FAKULTÄT FÜR INFORMATIK

Evolutionäre Algorithmen
Kapitel 1: Einführung

Sanaz Mostaghim

Intelligente Systeme

Institut für Wissens- und Sprachverarbeitung (IWS)

SS 2016

Inhalt und Lernziele

Übersicht

1. Einleitung
2. Biologische Grundlagen
3. Grundlagen evolutionärer Algorithmen
    1. Grundbegriffe
    2. Elemente
    3. Formale Definitionen

Lernziele

1. Die Welt von naturinspirierten Algorithmen kennenzulernen
2. Formulierung von Optimierungsproblemen
3. Formale Definitionen zu verstehen
4. Überblick über Elemente und Grundlagen evolutionärer Algorithmen bekommen

Warm Up!

In dieser Vorlesung geht es um Problemlösung!

Warm-Up Beispiel: Mit nur 3 Schritten wandeln Sie das Bild 1 in Bild 2

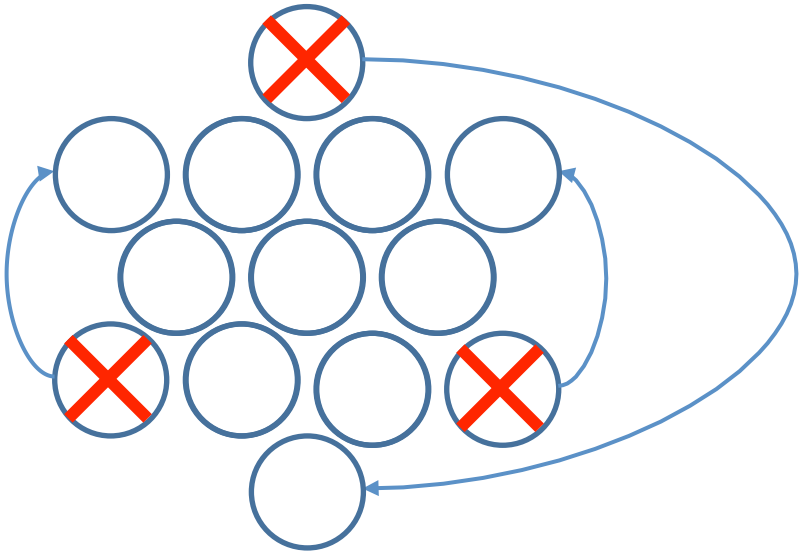


Bild 1

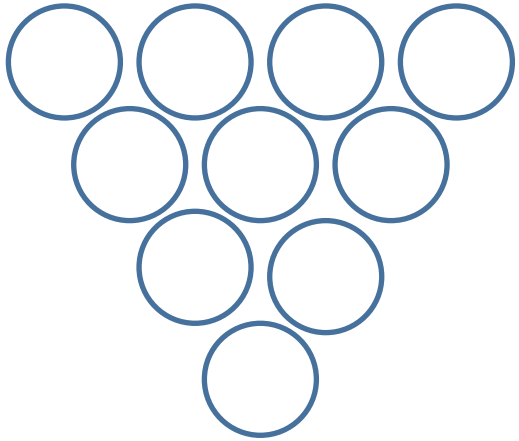


Bild 2

Warm Up!

- Finde die Alter von John‘s Kinder!

- What are the ages of my three sons?
- All of them celebrate their birthday today.
- The product of their ages is 36.
- The sum of their ages is equal to the number of the windows in that building.
- My oldest son has blue eyes.

36 x 1 x 1
18 x 2 x 1
12 x 3 x 1
9 x 4 x 1
9 x 2 x 2
6 x 6 x 1
6 x 3 x 2
4 x 3 x 3

36 + 1 + 1 = 38
18 + 2 + 1 = 21
12 + 3 + 1 = 16
9 + 4 + 1 = 14
9 + 2 + 2 = 13
6 + 6 + 1 = 13
6 + 3 + 2 = 11
4 + 3 + 3 = 10

9 + 2 + 2 = 13
6 + 6 + 1 = 13

Schwierigkeiten bei der Problemlösung

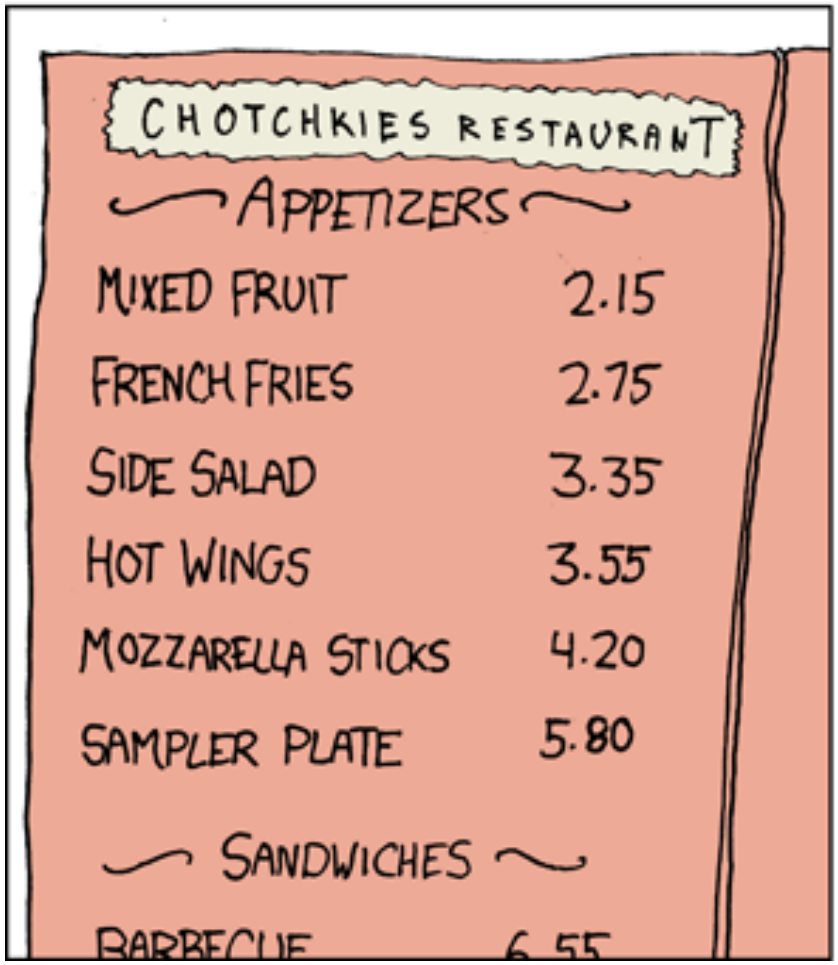
- Größe des Suchraums
  - TSP: n Städte -> (n-1)!/2 Optionen
  - SAT: n Parameter -> 2^n Optionen
- Modellierung des Problems
- Beschränkungen in Programmeirung

In dieser Vorlesung wollen wir die NP-schweren Probleme behandeln.
Was ist NP-schwer/hard?

NP bedeutet also:

Man kann nichtdeterministisch eine potentielle Lösung raten und dann in polynomieller Zeit die Korrektheit der Lösung überprüfen.

MY HOBBY:
EMBEDDING NP-COMPLETE PROBLEMS IN RESTAURANT ORDERS



From http://xkcd.com/287/

NP-vollständige Probleme

Der Begriff der NP-Vollständigkeit wurde 1971 von Stephen Cook definiert, seitdem wurden zahlreiche Probleme als NP-vollständig nachgewiesen, insbesondere

- SAT:
  - Erfüllbarkeitsproblem der Aussagenlogik (Entscheide für eine beliebige Formel der Aussagenlogik (in konjunktiver Normalform), ob sie erfüllbar ist oder nicht).
  - Das erste Problem, für das die NP-Vollständigkeit nachgewiesen wurde (von Cook)

- Traveling Salesperson Problem (TSP): Bestimme in einem beliebigen gewichteten Graphen eine Rundreise kürzester Länge (bzw. kleinsten Gewichts).

- Projektplanung mit beschränkten Ressourcen und zeitlichen Mindest- und Maximalabständen. (Hier ist bereits die Bestimmung einer zulässigen Lösung NP-vollständig!)

Einordnung: Computational Intelligence

Computational Intelligence =

Neuronale Netze (im Sommer)

+ Fuzzy-Systeme (im Winter)

+ Evolutionäre Algorithmen

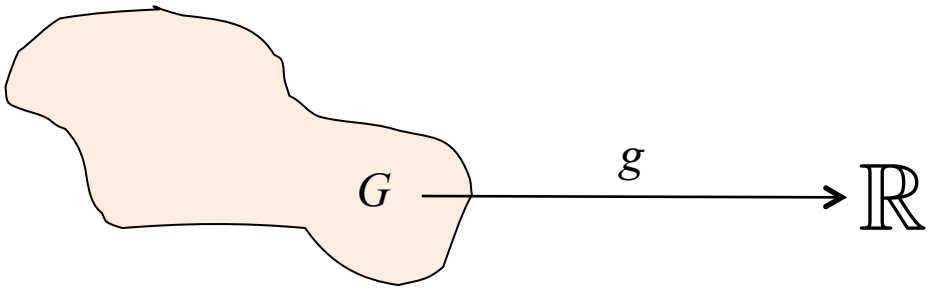
Computational Intelligence ist charakterisiert durch:

- meist „modellfreie“ Ansätze
- Approximation statt exakte Lösung
- schnelles Finden einer brauchbaren Lösung

Lösen von Optimierungsproblemen

Definition (Optimierungsproblem)

Ein Optimierungsproblem $(G, g, \succ)$ ist gegeben durch einen Lösungsraum G, eine Bewertungsfunktion $g : G \rightarrow \Re$, die jedem Lösungskandidaten einen Gütewert zuweist, sowie einer Vergleichsrelation $\succ \in \{<, >\}$



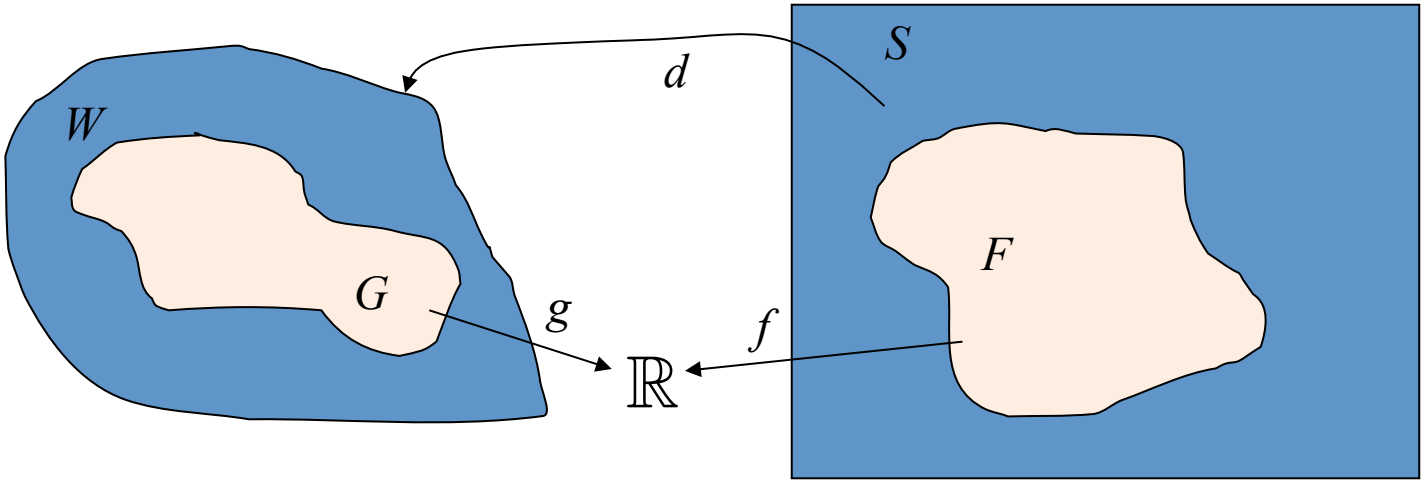
Dann ist die Menge der globalen Optima (Minima) X

$$X = \{x \in G | \forall x' \in G : g(x) \leq g(x')\}$$

Lösen von Optimierungsproblemen

Definition (Suchraum)

Der Suchraum S eines Problems ist der Raum, in dem ein Optimierungsalgorithmus $f : S \rightarrow \mathbb{R}$ suchen kann. Normalerweise sind der Lösungsraum und der Suchraum durch eine Kodierungsfunktion verbunden: $d : S \rightarrow W \supseteq G$



- gesucht: ein Element von X^*, das die Funktion f (global) optimiert (minimiert)

$$X^* = \{x \in S | \forall x' \in S : f(x) \leq f(x')\}$$

Optimierungsprobleme I

- Parameteroptimierung
  - z.B. Krümmung von Rohren für minimalen Widerstand
  - allgemein: Finden eines Parametersatzes, sodaß gegebene reellwertige Funktion ein (möglichst globales) Optimum annimmt
- Packprobleme
  - z.B. Füllen eines Rucksacks mit maximalem Wert
  - oder Packen möglichst weniger Kisten mit gegebenen Gütern
- Wegeprobleme
  - z.B. Problem des Handlungsreisenden (z.B. Bohren von Platinen)
  - Reihenfolge anzufahrender Ziele, Fahrtroutenoptimierung, Verlegen von Leiterbahnen auf Platinen/integrierten Schaltkreisen

Optimierungsprobleme II

- Anordnungsprobleme
  - z.B. Steinerproblem (engl. facility allocation problem):
  - Positionierung von Verteilerknoten z.B. in einem Telefonnetz
- Planungsprobleme
  - z.B. Ablaufpläne (Scheduling), Arbeitspläne, Operationenfolgen
  - (auch Optimierung in Compilern – Unordnung der Befehle)
- Strategieprobleme
  - z.B. Gefangenendilemma und andere Modelle der Spieltheorie
  - Verhaltensmodellierung von Akteuren im Wirtschaftsleben
- biologische Modellbildung
  - z.B. Netspinner (beschreibende Regeln zum Spinnennetz-Bau)
  - EA optimiert Parameter, Vergleich mit Realität -> gutes Modell

Modellierung -> Kodierung

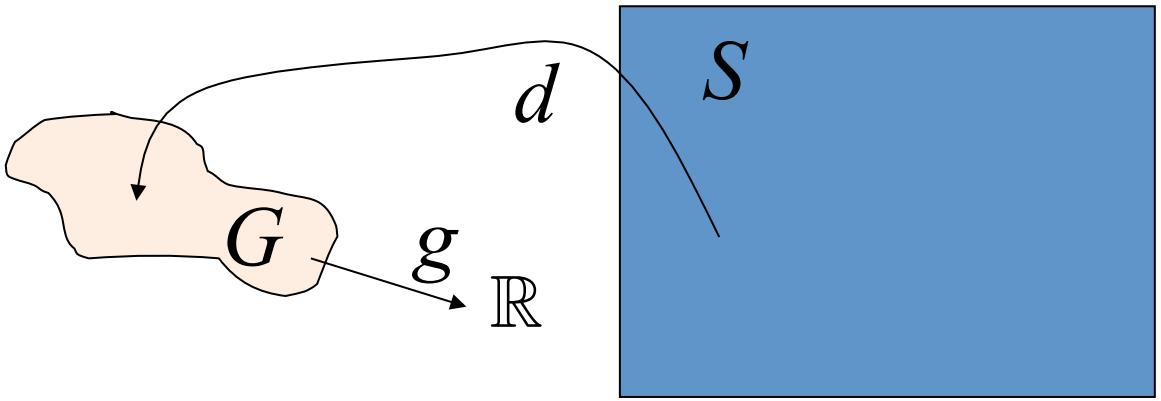
- Zur Lösung eines Optimierungsproblems:
  1. Modellierung des Suchraums und formale Beschreibung des Problems
  2. Auswahl eines geeignetes Algorithmus

Schritte für die Modellierung:

1. Auswahl einer Repräsentation -> Suchraum
2. Auswahl einer Nachbarschaftsfunktion
3. Auswahl eine Fitnessfunktion (Gütefunktion) -> Zeigt wie gut eine Lösung ist.

Repräsentation

- Die Repräsentation definiert den Suchraum S
- Der Suchraum muss das globale Optimum beinhalten!



Beispiel: Finde die optimal Lösung für

$$g(y) = y^2, G = \{y \in \Re | y \in [-2, 2]\}$$

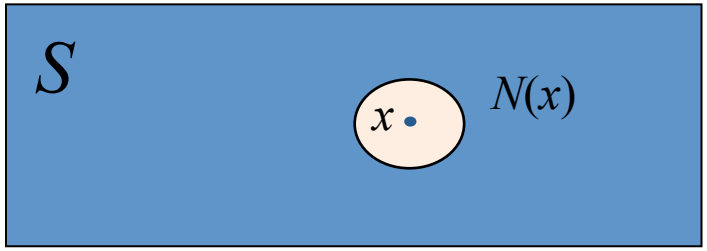
Repräsentation: Binär mit 3 Bits -> die optimale Lösung wird nie erreicht, warum? -> Übung

Eigenschaften einer Repräsentation

- Die Repräsentation soll in Zusammenhang mit einer Nachbarschaftsfunktion ausgewählt werden.
- In dem Suchraum sollen alle Lösungen erreichbar sein!
- Eine Bewegung (Move) in der Nachbarschaft einer Lösung soll die wichtige Charakteristiken der Lösung bewahren ("locality")
- Die Lösungen in einer Nachbarschaft sollen ähnliche Qualität (Güte oder Fitness) haben.

Die **Nachbarschaftsfunktion** ist eine Abbildung $N : S \rightarrow \wp(S)$, die für jede Lösung $x \in S$ eine Menge $N(x) \subseteq S$ von Lösungen zuordnet.

Die Menge $N(x)$ ist die Nachbarschaft der Lösung x, und $y \in N(x)$ ist ein Nachbar von x mit der Annahme: $x \in N(x),\ \forall x \in S.$



Repräsentationen

Typische Repräsentationen:

- Binär, Reelle Vektoren, Permutationen, Integer-Kette, ...

Binary (Binär): $x \in B^n$

1. Nachbarschaftsfunktion: „Flip one Bit“

- Deterministische Nachbarschaftsfunktion: $N(x) = \{y \mid H(x,y) \leq 1\}$
- H(x,y) - > Hamming-Abstand
- Beispiel: x = 00100 -> N(x) = {(00101), (00110), (00000), (01100), (10100)}

2. Nachbarschaftsfunktion: Inverse –Mutation

- Inversion-Mutation: Für k = 1, …, n, flip bits von k bis n
- Beispiel: x = 001000 -> N(00100) = {(11011),(01011),(00011),(00111), (00101)}

Typische Optimierungsprobleme I

SAT: Erfüllbarkeitsproblem

Ziel: ob eine in konjunktiver Normalform vorliegende aussagenlogische Formel f, die höchstens 3 Literale x_i pro Klausel $\{C_1, C_2, \ldots C_m\}$ enthält, erfüllbar ist:

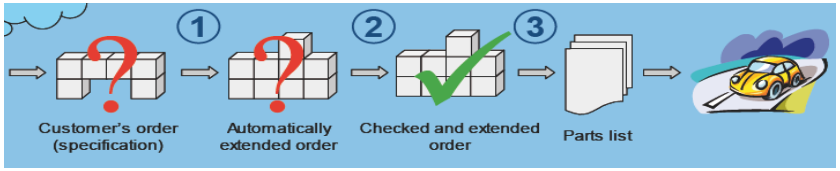
$$f = C_1 \wedge C_2 \wedge \ldots \wedge C_m$$

Gesucht ist nun eine Belegung der Variablen $x = \{x_1, x_2, \ldots, x_n\}, x_i \in \{0,1\}$, für die f den Wert 1 (wahr) annimmt. Falls es eine solche Belegung gibt, ist f erfüllbar, sonst nicht.

Modellierung:

- Repräsentation: Binär
- Nachbarschaftsfunktion für binäre Repräsentation
- Gütefunktion: Anzahl der Klausel, die wahr sind

Anwendung: Order Processing



Typische Optimierungsprobleme II

Rucksack Problem

- Gegeben: N Objekte mit unterschiedlichen Gewichte W_{oi}
- Ziel: Füllen eines Rucksacks mit maximalem Kapazität W_c

Modellierung:

- Binäre Repräsentation mit Länge = N

- > 1: Objekt im Rucksack, 0: nicht im Rucksack

Obj_1	Obj_2	Obj_3	…		Obj_{N-1}	Obj_N

- Nachbarschaftsfunktion für binäre Repräsentation
- Gütefunktion: Anzahl der Objekte im Rucksack und die Summe deren Gewichte

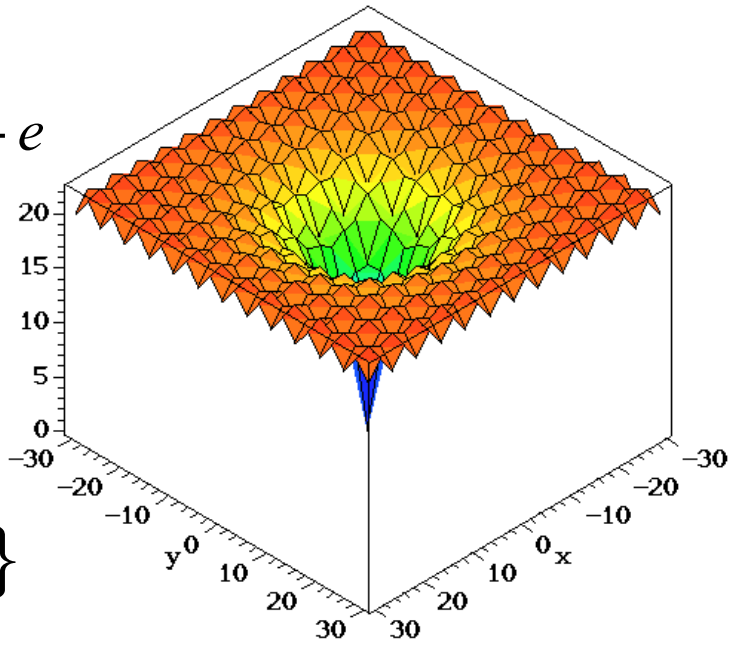
Typische Optimierungsprobleme III

Parameteroptimierung

n-dimensionaler Suchraum von reelle vektoren $\vec{x} \in \Re^n$

$$\min f(\vec{x}) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_n x_i^2}} - e^{0.5\sum_n \cos(2\pi x_i)} + 20 + e$$

$$s.t. \quad -30.0 \le x_i \le 30.0$$



<u>Modellierung:</u>

- Repräsentation -> reelle Vektoren
- Nachbarschaftsfunktion:

$$N(\vec{x}) = \{\vec{y} | d(\vec{x}, \vec{y}) \le \epsilon\}$$

$d(\vec{x}, \vec{y})$-> Euklidischer Abstand

  - Input: $\vec{x}$
  - Output: $\vec{y} = \vec{x} + \vec{v}$

    $\vec{v}$ ist ein Zufallsvektor

- Gütefunktion: Funktionswert

Typische Optimierungsprobleme IV

Bin Packing Problem

- Gegeben: N Objekte mit unterschiedlichen Gewichten W_{oi}
- Ziel: Verteilung von N Objekte auf B Bins mit jeweils C_{Bi} als Kapazität, so dass alle Objekte gepackt sind
- Anwendung: Transport, Software Engineering, Planungsprobleme, etc.

Modellierung 1:

- Repräsentation: Integer-Kette mit Länge N, jede Element ist eine Bin-Zahl

Obj_1	Obj_2	Obj_3	…		Obj_{N-1}	Obj_N

Beispiel: (2 3 4 5 4 1 4 1 2 2), für B = 5 und N = 10

- Nachbarschaftsfunktion:
  - Austausch von zwei Objekten (2 3 1 5 4 4 4 1 2 2)
  - Zufällige Zuordnung eines Objekts zu einem anderen Bin (2 3 4 5 3 1 4 1 2 2)

Typische Optimierungsprobleme V

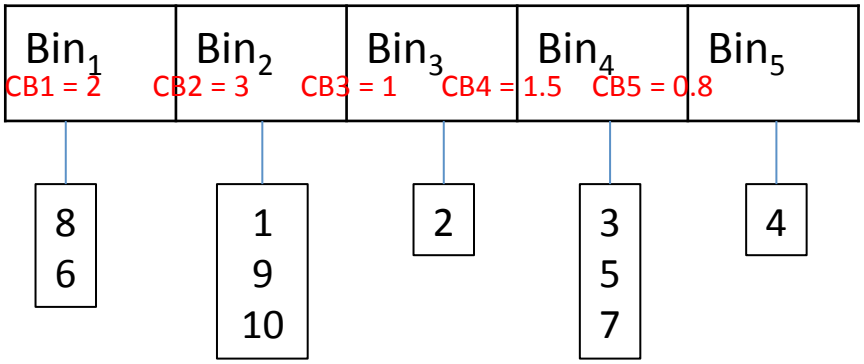
Modellierung 2:

- Repräsentation: Matrix von Integer-Werte

  NxB Matrix

  Beispiel:

8 1 2 3 4
6 9 0 5 0
0 10 0 7 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0



- Nachbarschaftsfunktion: Austausch von zufälligen Objekte in zwei Spalten

Typische Optimierungsprobleme VI

TSP (Traveling Salesman Problem)

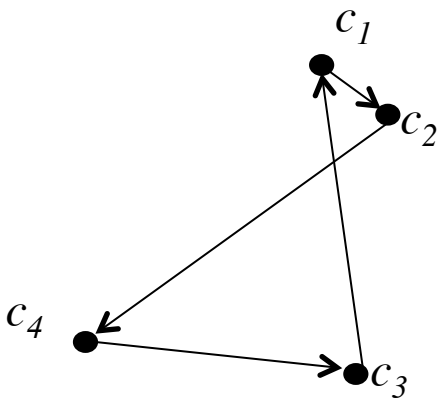
- Gegeben: eine Menge von N Städte $C=\{c_1,..c_N\}$ und eine Abstandsfunktion $d : C \times C \rightarrow \Re$, die den Abstand zwischen zwei Städte c_i und c_j berechnet.

- Ziel: Eine Permutation π von Städte zu finden, die den Abstand L Zwischen c_1 und c_N minimiert:

$$L = \sum_{i=1}^{N-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(N)}, c_{\pi(1)})$$

Beispiel:

$\pi = (1, 2, 4, 3)$



Typische Optimierungsprobleme VII

Modellierung 1:

- Repräsentation: Permutation , Beispiel: (1 4 6 5 2 3), für N = 6
- Nachbarschaftsfunktion:
  - Umtausch von zwei Städte: 1-4-6-5-2-3 → 1-**2**-6-5-**4**-3
  - Lösche eine Stadt und füge in eine andre Stelle ein

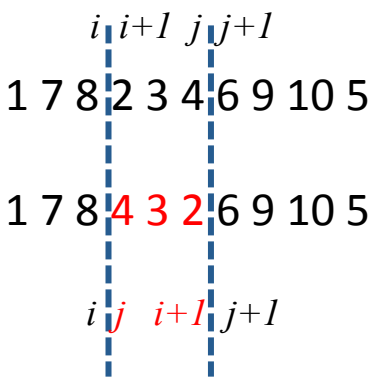
    1-4-6-5-2-3 → 1-6-5-2-**4**-3
  - Swap: Wähle eine zufällige Sub-Tour und swap die Ordnung der Städte

    1-4-6-5-2-3 → 1-4-**2-5-6**-3

    Vorteil von Swap: Effiziente Evaluation von L, für das Swap zwischen i und j:

$$L_{new} = L_{old} - d(i,i+1) - d(j,j+1) + d(i+1,j+1) + d(i,j)$$

$$L_{new} = L_{old} - d(8,2) - d(4,6) + d(2,6) + d(8,4)$$



Typische Optimierungsprobleme VIII

Modellierung 2:

- Repräsentation: Random key encoding – reelle Vektor Länge N von Prioritätswerte für eine Permutation, Beispiel für N = 6, (0.3, 0.2, 0.7, 0.4, 0.9, 0.5)

$$\left.\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ 0.3 & 0.2 & 0.7 & 0.4 & 0.9 & 0.5 \end{array}\right\} \rightarrow 5-3-6-4-1-2$$

- Nachbarschaftsfunktion: Standard für reelle Vektoren (gaussian noise)

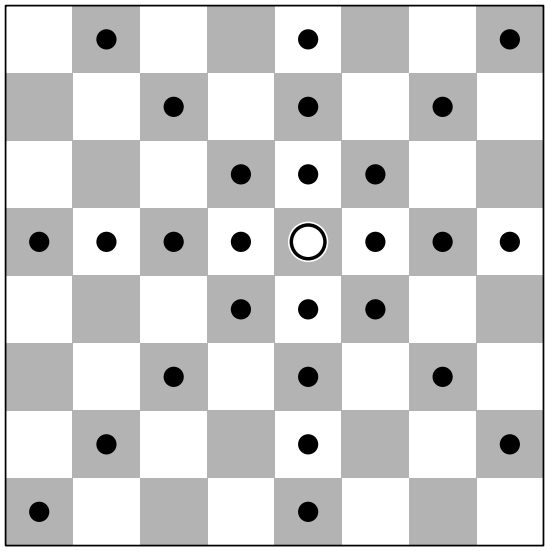
Modellierung 3:

- Repräsentation: Jump – Integer-Ketten Länge N von „Jumps“, die Reihenfolge von Städte zeigen, Beispiel: (1,3,2,4,2,1)
- Nachbarschatfsfunktion: Zuffälige Änderung eines Integerwerts

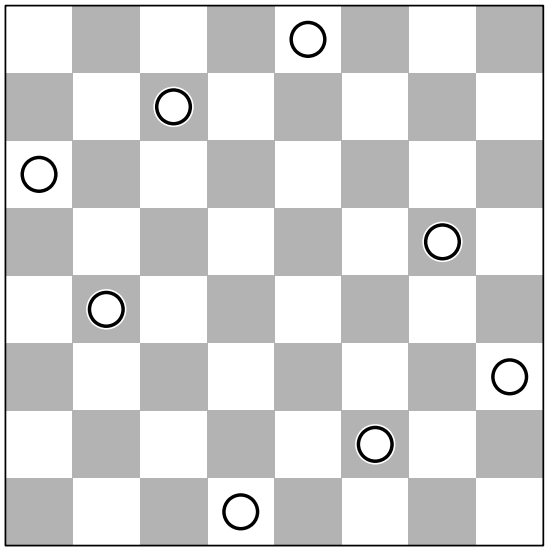
1,2,3,4,5,6 → 1-
1,2,3,4,5,6 → 1-4-
1,2,3,**4**,5,6 → 1-4-6-
1,2,3,**4**,5,**6** → 1-4-6-2-
1,2,3,**4**,5,**6** → 1-4-6-2-5-
1,2,3,**4,5,6** → 1-4-6-2-5-3

Das n-Damen-Problem I

- Platziere n Damen auf n x n Schachbrett, sodass in keiner Reihe, keiner Linie und keiner Diagonale mehr als eine Dame ist
- oder: Platziere Damen, sodass keine Dame einer anderen im Weg ist



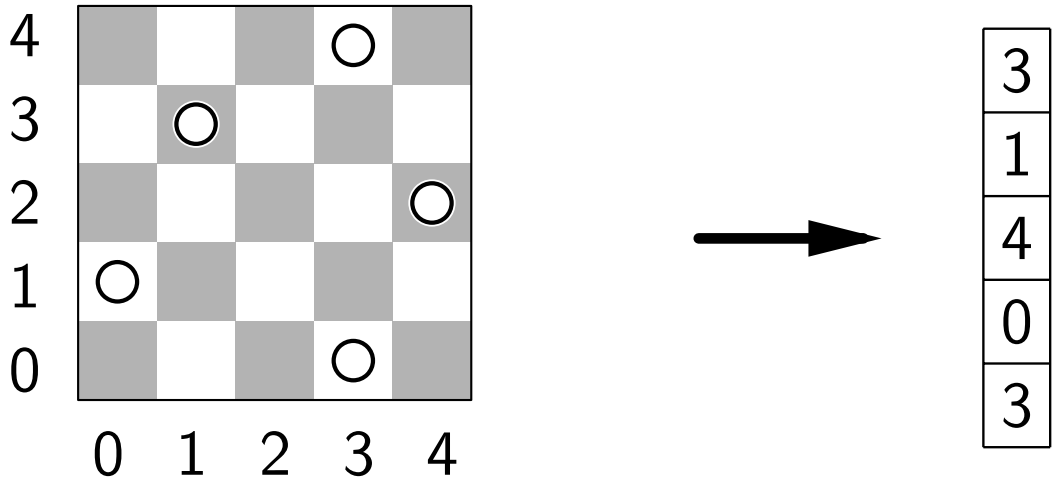
Zugmöglichkeiten einer Dame



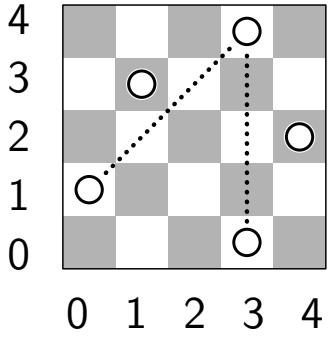
Lösung des 8-Damen-Problems

Das n-Damen-Problem II

- Repräsentation: Integer-Kette mit n Elementen, die die Position der Dame in zugehöriger Zeile zeigen



- Gütefunktion: negierte Zahl der Spalten und Diagonalen mit 1 Dame (negierte Zahl, weil zu maximierende Fitness)



2 Kollisionen $\rightarrow$ Fitness $= -2$

Prinzipielle Lösungsansätze

analytische Lösung:

- sehr effizient, aber nur seltenen anwendbar

vollständige Durchforstung:

- sehr ineffizient, daher nur bei sehr kleinen Suchräumen anwendbar

blinde Zufallssuche:

- immer anwendbar, aber meist sehr ineffizient

gesteuerte Suche:

- Voraussetzung: Funktionswerte ähnlicher Elemente aus Suchraum S sind sich ähnlich

- > Evolutionäre Algorithmen (EAs) werden oft zum Lösen von Optimierungsproblemen genutzt.

EAs: Biologische Grundlagen

- Evolutionäre Algorithmen (EAs) basieren auf **biolog. Evolutionstheorie** [Darwin, 1859].
- empfehlenswert: [Dawkins, 1986, Dawkins, 1989] (Englisch),

- grundsätzliches Prinzip:
  - **Durch zufällige Variation entstehende, vorteilhafte Eigenschaften werden durch natürliche Auslese ausgewählt.**
  - Survival of the fittest! Individuen mit vorteilhaften Eigenschaften haben bessere Fortpflanzungs- und Vermehrungschancen – *„differentielle Reproduktion“*
- Evolutionstheorie erklärt Vielfalt und Komplexität der Lebewesen
- erlaubt Vereinigung aller Disziplinen der Biologie

Prinzipien der organismischen Evolution I

nach [Vollmer, 1995]

Diversität

- alle Lebewesen (sogar in derselben Art) sind *verschieden*
- jedes Erbgut ist anders -> *Vielfalt des Lebens*
- jetzt existierende Lebewesen = winziger Bruchteil aller möglichen

Variation

- *neue Varianten* durch Mutation und genetische Rekombination (sexuelle Fortpflanzung)

Vererbung

- Variationen sind *erblich*, wenn sie in Keimbahn gelangen
- werden also genetisch an nächste Generation weitergegeben
- i.A. keine Vererbung erworbener Eigenschaften (*Lamarckismus*)

Prinzipien der organismischen Evolution II

Artbildung

- *genetische Divergenz* von Individuen und Populationen
- d.h. neue *Arten* (Indiv. nicht fruchtbar miteinander kreuzbar)
- charakt. Verzweigungen des phylogenet. (stammesgesch.) Baumes

Überproduktion, fast alle Lebewesen:

- *nicht alle Nachkommen* können Reproduktionsreife erreichen

Anpassung / natürliche Auslese / differentielle Reproduktion

- im Durchschnitt: erbliche Variationen der Überlebenden einer Pop.
- d.h. *erhöht Anpassung* an lokale Umgebung
- Herbert Spencers Slogan “survival of the fittest” ist irreführend
- besser „unterschiedl. Tauglichkeit -> unterschiedl. Vermehrung“

Prinzipien der organismischen Evolution III

Zufälligkeit / blinde Variation

- Variationen werden *zufällig ausgelöst/bewirkt/verursacht*
- keine Ausrichtung auf best. Merkmale/günstige Anpassungen

Gradualismus

- Variationen erfolgen in vergleichsweise *kleinen Stufen* (gemessen am gesamten Informationsgehalt/Komplexität des Organismus)
- d.h. phylogenetische Veränderungen = *graduell* und langsam

Evolution / Transmutation / Vererbung mit Modifikation

- Anpassung an Umgebung -> Arten *entwickeln* sich allmählich
- Evolutionstheorie im Gegensatz zum Kreationismus (Behauptung der Unveränderlichkeit der Arten)

Prinzipien der organismischen Evolution IV

diskrete genetische Einheiten

- Speichern/Übertragen/Ändern des Erbguts in diskreten Einheiten
- keine kontinuierliche Verschmelzung von Erbmerkmalen
- ansonsten: durch Rekombination zum *Jenkins nightmare* (völliges Verschwinden jeglicher Verschiedenheit in Population)

Opportunismus

- evolutive Prozesse arbeiten nur mit dem, was vorhanden ist
- nicht mit dem, was es einmal gab oder geben könnte
- bessere/optimale Lösungen werden nicht gefunden, wenn benötigte evolutiven Zwischenstadien „schlechter“ wären

Prinzipien der organismischen Evolution V

evolutionsstrategische Prinzipien

- nicht nur Organismen werden optimiert, sondern auch Mechanismen der Evolution: Vermehrungs- und Sterberaten, Lebensdauern, Anfälligkeit gegenüber Mutationen, Mutationsschrittweiten, Evolutionsgeschwindigkeit, etc.

ökologische Nischen

- konkurrierende Arten können einander tolerieren, wenn sie versch. Ökonischen („Lebensräume“) besetzen/schaffen
- trotz Konkurrenz und natürlicher Auslese: Artenvielfalt möglich

Irreversibilität

- Lauf der Evolution ist irreversibel und unwiederholbar

Prinzipien der organismischen Evolution VI

Nichtvorhersagbarkeit

- Lauf der Evolution:
  - nicht determiniert,
  - nicht programmiert,
  - nicht zielgerichtet

→ nicht vorhersagbar

wachsende Komplexität

- biologische Evolution führt i.A. zu immer komplexeren Systemen
- Problem: Wie misst man Komplexität von Lebewesen?

Grundlagen evolutionärer Algorithmen

Grundbegriffe und ihre Bedeutung

Begriff	Biologie	Informatik
Individuum	Lebewesen	Lösungskandidat
Chromosom legt „Bauplan“ bzw. (Teil der Eigenschaften) eines Individuums in kodierter Form fest	DNS-Histon-Protein-Strang - meist mehrere Chromsomen je Individuum	Zeichenkette - meist nur ein Chromosom je Individuum
Gen grundlegende Einheit der Vererbung, die eine (Teil-)Eigenschaft eines Individuums festlegt	Teilstück eines Chromosoms	ein Zeichen
Allel (Allelomorph) je Chromosom gibt es nur eine Ausprägung eines Gens	Ausprägung eines Gens	Wert eines Zeichens

Grundbegriffe und ihre Bedeutung

Begriff	Biologie	Informatik
Locus in einem Chromosom gibt es an jedem Ort genau ein Gen	Ort eines Gens	Position eines Zeichens
Phänotyp	äußeres Erscheinungsbild eines Lebewesens	Umsetzung/ Implementierung eines Lösungskandidaten
Genotyp	genetische Konstitution eines Lebewesens	Familie/Multimenge von Chromosomen
Population	Menge von Lebewesens	Wert eines Zeichens
Generation	Population zu einem Zeitpunkt	
Reproduktion	Erzeugung von Nachkommen aus einem oder mehreren (meist zwei) Lebewesen	Erzeugen von (Kind-)Chromosomen aus 1 oder mehreren (Eltern-)Chromosomen
Fitness bestimmt Überlebens- und Fortpflanzungschancen	Tauglichkeit eines Lebewesens	Güte/Tauglichkeit eines Lösungskandidaten

Elemente eines evolutionären Algorithmus I

Kodierungsvorschrift für Lösungskandidaten

- problemspezifische Kodierung der Lösungskandidaten
- keine allgemeinen Regeln
- später: einige Aspekte, zur Beachtung bei Wahl einer Kodierung

Methode, die **Anfangspopulation** erzeugt

- meistens Erzeugen zufälliger Zeichenketten
- auch komplexere Verfahren je nach gewählter Kodierung

Bewertungsfunktion (Fitnessfunktion) für die Individuen

- stellt Umgebung dar und gibt Güte der Individuen an
- meist identisch mit zu optimierender Funktion
- enthält auch zusätzliche Elemente (z.B. Nebenbedingungen)

Elemente eines evolutionären Algorithmus II

Auswahlmethode basierend auf Fitnessfunktion

- bestimmt Individuen für Erzeugung von Nachkommen
- wählt Individuen (unverändert) in nächste Generation

genetischen Operatoren, die Lösungskandidaten ändern

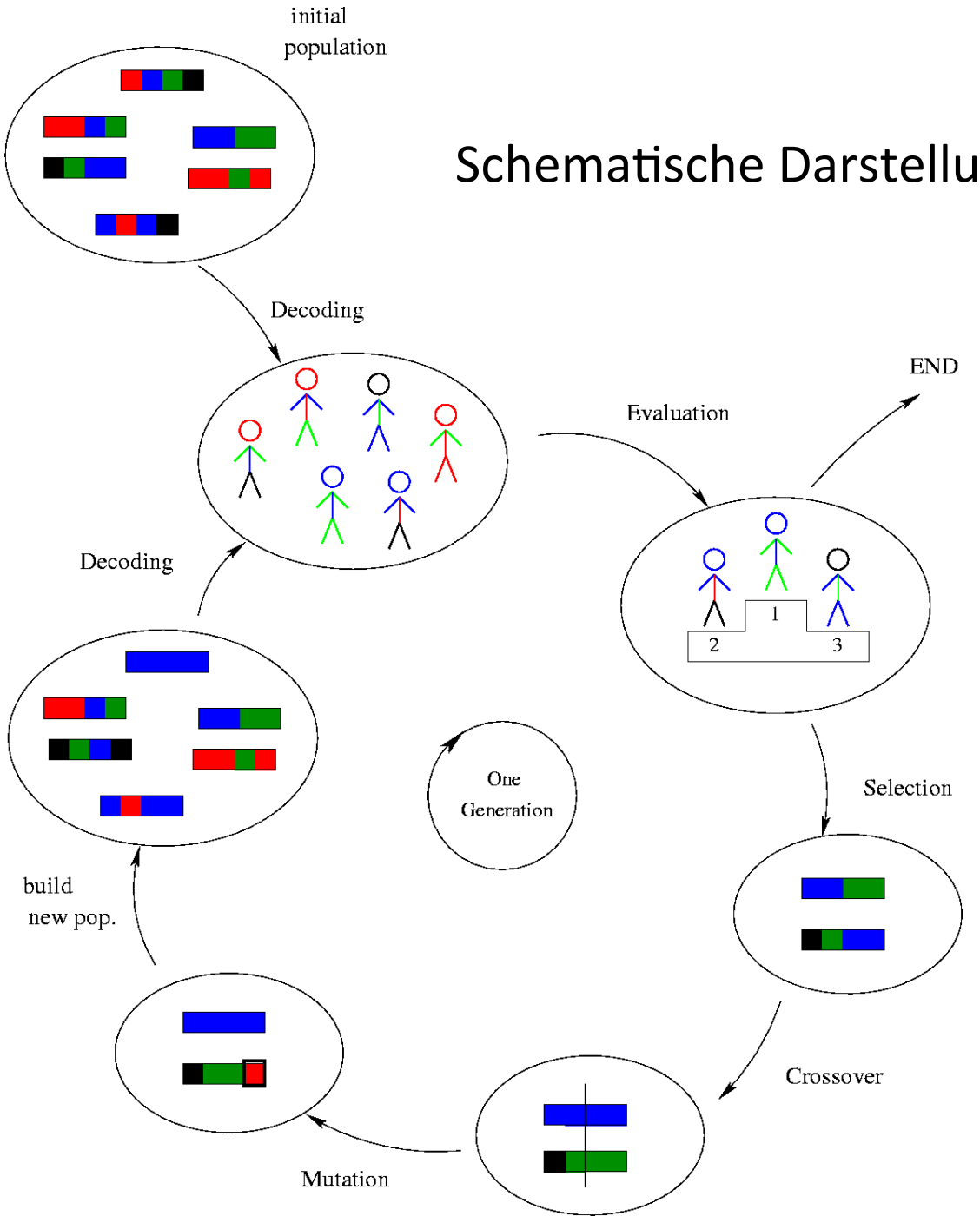
- *Mutation* — zufällige Veränderung einzelner Gene
- *Crossover* — Rekombination von Chromosomen
  - Chromsomen werden zerteilt und dann überkreuzt zusammengefügt

Parameterwerte (Populationsgröße, Mutationsw'keit, etc.)

Abbruchkriterium, z.B.

- festgelegte Anzahl von Generationen berechnet
- keine Verbesserung
- vorgegebene Mindestlösungsgüte erreicht

Schematische Darstellung



Formale Definitionen: Dekodierungsfunktion

nach [Weicker, 2007]

- für jedes Optimierungsproblem: unterschiedliche Darstellung der Lösungskandidaten (z. B. Folien EA-1-17 – EA-1-25)

- EAs trennen Suchraum S (sog. Phänotyp) von Darstellung des Lösungskandidaten in Individuum (sog. Genotyp $\mathcal{G}$)

- Bewertungsfunktion *f* ist definiert auf S
- Mutation und Rekombination sind auf $\mathcal{G}$ definiert
- Bewertung eines im Genotyp vorliegenden Individuums durch Abbildung in S

Definition (Dekodierungsfunktion)

Eine Dekodierungsfunktion $dec : \mathcal{G} \rightarrow S$ ist eine Abbildung vom Genotyp $\mathcal{G}$ auf den Phänotyp S.

Individuum

Individuum besteht i.A. aus drei Dingen:

1. *Genotyp* $A.G \in \mathcal{G}$ eines Individuums *A*
2. *Zusatzinformationen* oder *Strategieparameter* $A.S \in \mathcal{Z}$
   - z.B. Parametereinstellungen für Operatoren
   - Raum $\mathcal{Z}$ aller möglichen Zusatzinformationen
   - *A.S* sind ebenso wie *A.G* durch Operatoren modifizierbar
3. *Güte* oder *Fitness A.F*

Definition (Individuum)

- Ein *Individuum A* ist ein Tupel (*A.G*,*A.S*,*A.F*) bestehend aus dem eigentlichen Lösungskandidaten (dem Genotyp $A.G \in \mathcal{G}$), den optionalen Zusatzinformationen $A.S \in \mathcal{Z}$ und dem Gütewert $A.F = f(\mathrm{dec}(A.G))$.

Operatoren

Definition (Operatoren)

Für ein durch $\mathcal{G}$ kodiertes Optimierungsproblem und $\mathcal{Z}$ wird ein *Mutationsoperator* definiert durch die Abbildung

$$Mut^{\xi} : \mathcal{G} \times \mathcal{Z} \to \mathcal{G} \times \mathcal{Z}$$

Analog wird ein *Rekombinationsoperator* mit $r \geq 2$ Eltern und $s \geq 1$ Kindern ($r, s \in \mathbb{N}$) definiert durch die Abbildung

$$Rek^{\xi} : (\mathcal{G} \times \mathcal{Z})^r \to (\mathcal{G} \times \mathcal{Z})^s$$

Selektionsoperator

- Eingabe: Population mit *r* Individuen, wobei *s* gewählt werden
- Selektion verändert/erfindet keine neuen Individuen
- Selektion bestimmt Indizes der Individuen nur durch Gütewerte

Definition (Selektionsoperator)

- Ein *Selektionsoperator* Sel wird auf eine Population

$P = \langle A^{(1)}, \ldots, A^{(r)} \rangle$ angewandt:

$$Sel^{\xi} : (\mathcal{G} \times \mathcal{Z} \times \mathbb{R})^r \to (\mathcal{G} \times \mathcal{Z} \times \mathbb{R})^s$$

$$\langle A^{(i)} \rangle_{1 \leq i \leq r} \to \langle A^{IS^{\xi}(c_1,\ldots,c_r)_k} \rangle_{1 \leq k \leq s}$$

Die dabei zugrunde gelegte Indexselektion hat die Form

$$IS^{\xi} : \mathbb{R}^r \to \{1, \ldots, r\}^s$$

Beispiel zum Selektionsoperator

- Elternpopulation bestehe aus Individuen $A(1),A(2),...,A(5)$
- jeweiligen Gütewerte der Individuen seien

1. $A(1).F = 2.5$
2. $A(2).F = 1.9$
3. $A(3).F = 3.7$
4. $A(4).F = 4.1$
5. $A(5).F = 2.4$

- Selektion wählt mit $IS^{\xi} : \mathbb{R}^5 \rightarrow \{1, ..., 5\}^3$ Indizes 4, 3 und 1 bzw. Individuen $A(4)$, $A(3)$ und $A(1)$

Generischer Grundalgorithmus

Definition

Ein *einfacher evolutionärer Algorithmus* zu einem Optimierungsproblem $(S, f, \succ)$ ist ein 8-Tupel

$(\mathcal{G}, dec, Mut, Rek, IS_{Eltern}, IS_{Umwelt}, \mu, \lambda)$

Dabei bezeichnen μ die Anzahl an Individuen in der Elternpopulation und λ die Anzahl der erzeugten Kinder pro Generation. Ferner gelten

$$Rek : (\mathcal{G} \times \mathcal{Z})^k \rightarrow (\mathcal{G} \times \mathcal{Z})^{k\prime}$$

$$IS_{Eltern} : \mathbb{R}^\mu \rightarrow (1, ..., \mu)^{\frac{k}{k'} \cdot \lambda} \qquad \frac{k}{k'} \cdot \lambda \in \mathbb{N}$$

$$IS_{Umwelt} : \mathbb{R}^{\mu+\lambda} \rightarrow (1, ..., \mu + \lambda)^\mu$$

Generischer Grundalgorithmus

Algorithm 1 EA-Schema

Input: Optimierungsproblem $(S, f, \succ)$

$t \leftarrow 0$
pop$(t) \leftarrow$ erzeuge Population der Größe μ
bewerte pop(t)
while Terminierungsbedingung nicht erfüllt $\{$
  pop$_1 \leftarrow$ selektiere Eltern für λ Nachkommen aus pop(t)
  pop$_2 \leftarrow$ erzeuge Nachkommen durch Rekombination aus pop$_1$
  pop$_3 \leftarrow$ mutiere die Individuen in pop$_2$
  bewerte pop$_3$
  $t \leftarrow t + 1$
  pop$(t) \leftarrow$ selektiere μ Individuen aus pop$_3 \cup$ pop$(t-1)$
$\}$
return bestes Individuum aus pop(t)