Genetische Algorithmen

Christian Borgelt

Arbeitsgruppe Neuronale Netze und Fuzzy-Systeme
Institut für Wissens- und Sprachverarbeitung
Otto-von-Guericke-Universität Magdeburg
Universitätsplatz 2, D-39106 Magdeburg

borgelt@iws.cs.uni-magdeburg.de
http://fuzzy.cs.uni-magdeburg.de/∼borgelt/

http://fuzzy.cs.uni-magdeburg.de/studium/ga/

Genetische Algorithmen: Einleitung

Einordnung: Soft Computing

Soft Computing	=	**neuronale Netze**	(Parallelvorlesung)
	+	**Fuzzy-Systeme**	(im Wintersemester)
	+	**genetische Algorithmen**	

Soft Computing ist charakterisiert durch:

- Meist „modellfreie“ Ansätze
  (d.h., es ist kein explizites Modell des zu beschreibenden Gegenstandbereichs notwendig; „modellbasiert“ dagegen: z.B. Lösen von Differentialgleichungen)
- Approximation statt exakte Lösung (nicht immer ausreichend!)
- Schnelleres Finden einer brauchbaren Lösung,
  u.U. auch ohne tiefgehende Problemanalyse

Genetische Algorithmen: Anwendungsgebiete

Allgemein: **Lösen von Optimierungsproblemen**

- Gegeben:
  - ein Suchraum S
  - eine zu optimierende Funktion $f : S \to \mathbb{R}$
  - ggf. einzuhaltende Nebenbedingungen
- Gesucht: Ein Element $s \in S$, das die Funktion f optimiert.
- **Prinzipielle Lösungsansätze:**
  - *analytische Lösung:* sehr effizient, aber nur in seltenen Fällen 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 des Suchraums sind ähnlich.

Genetische Algorithmen: Anwendungsgebiete

Beispiele für Optimierungsprobleme

- **Parameteroptimierung**
  z.B. Krümmung von Rohren für minimalen Widerstand
  Allgemein: Finden eines Parametersatzes, so daß eine 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 (Anwendung: Bohren von Platinen)
  Reihenfolge von anzufahrenden Zielen, Fahrtroutenoptimierung
  Verlegen von Leiterbahnen auf Platinen und in integrierten Schaltkreisen

Genetische Algorithmen: Anwendungsgebiete

- **Anordnungsprobleme**

  z.B. Steinerproblem (facility allocation problem):
  Positionierung von Verteilerknoten z.B. in einem Telefonnetz

- **Planungsprobleme**

  z.B. Ablaufpläne (Scheduling), Arbeitspläne, Operationenfolgen
  (auch z.B. zur Optimierung in Compilern — Umordnung der Befehle)

- **Strategieprobleme**

  z.B. Gefangenendilemma und andere Modelle der Spieltheorie,
  Verhaltensmodellierung von Akteuren im Wirtschaftsleben

- **biologische Modellbildung**

  z.B. Netspinner (regelbasiertes Modell, das beschreibt, wie eine Spinne ihr Netz baut; Parameter werden durch einen genetischen Algorithmus optimiert und mit Beobachtungen verglichen; liefert recht gutes Modell)

Biologische Grundlagen

Genetische Algorithmen: Motivation

- Genetische Algorithmen basieren auf der **biologischen Evolutionstheorie**

  Charles R. Darwin: "On the Origin of Species by Means of Natural Selection" („Die Entstehung der Arten durch natürliche Zuchtwahl"), London 1859

  Empfehlenswerte Literatur zur biologischen Evolutionstheorie sind speziell die Bücher von Richard Dawkins, z.B. "The Selfish Gene" („Das egoistische Gen") und "The Blind Watchmaker" („Der blinde Uhrmacher").

- Grundsätzliches Prinzip:

  Durch zufällige Variation entstehende vorteilhafte Eigenschaften werden durch natürliche Auslese ausgewählt.

  (Individuen mit vorteilhaften Eigenschaften haben bessere Fortpflanzungs- und Vermehrungschancen — „differentielle Reproduktion".)

- Die Evolutionstheorie erklärt die Vielfalt und Komplexität der Lebewesen und erlaubt es, alle Disziplinen der Biologie zu vereinen.

Prinzipien der organismischen Evolution I

- **Diversität**

  Alle Lebewesen, sogar solche innerhalb ein und derselben Art, sind voneinander *verschieden*, und zwar bereits in ihrem Erbgut (*Vielfalt des Lebens*).

  Gleichwohl bilden die tatsächlich existerenden Formen von Lebewesen nur einen winzigen Bruchteil der im Prinzip möglichen.

- **Variation**

  Es entstehen, durch Mutation und genetische Rekombination (sexuelle Fortpflanzung), laufend *neue Varianten*.

- **Vererbung**

  Die Variationen sind, soweit sie in die Keimbahn gelangen, *erblich*, werden also genetisch an die nächste Generation weitergegeben.

  (i.a. keine Vererbung von erworbenen Eigenschaften — sog. *Lamarckismus*)

(nach Gerhard Vollmer: „Der wissenschaftstheoretische Status der Evolutionstheorie — Einwände und Gegenargumente" in: „Biophilosophie", Reclam, Stuttgart 1995)

Prinzipien der organismischen Evolution II

- **Artbildung**

  Es kommt zur *genetischen Divergenz* von Individuen und Populationen; es entstehen neue *Arten*, deren Vertreter nicht mehr fruchtbar miteinander kreuzbar sind. Die Artbildung verleiht dem phylogenetischen (stammesgeschichtlichen) „Stammbaum“ seine charakteristische Verzweigungsstruktur.

- **Überproduktion**

  Fast alle Lebewesen erzeugen *mehr Nachkommen*, als jemals zur Reproduktionsreife kommen können.

- **Anpassung / natürliche Auslese / differentielle Reproduktion**

  Im Durchschnitt weisen die Überlebenden einer Population solche erblichen Variationen auf, die ihre *Anpassung* an die lokale Umgebung *erhöhen*.

  Herbert Spencers Redewendung vom „Überleben der Tauglichsten“ (“survival of the fittest”) ist allerdings eher irreführend; besser spricht man von „unterschiedlicher Vermehrung aufgrund unterschiedlicher Tauglichkeit“.

Prinzipien der organismischen Evolution III

- **Zufälligkeit / blinde Variation**

  Variationen sind *zufällig*, zwar *ausgelöst*, *bewirkt*, *verursacht*, aber nicht vorzugsweise auf bestimmte Merkmale oder günstige Anpassungen ausgerichtet (*nicht teleologisch*, von griech.: $\tau\varepsilon\lambda o\varsigma$ — Ziel, Zweck).

- **Gradualismus**

  Variationen erfolgen in vergleichsweise *kleinen Stufen*, gemessen am gesamten Informationsgehalt oder an der Komplexität des Organismus. Deshalb sind phylogenetische (stammesgeschichtliche) Veränderungen *graduell* und relativ langsam. (Gegensatz: Saltationismus — große Entwicklungssprünge)

- **Evolution / Transmutation / Vererbung mit Modifikation**

  Wegen der Anpassung an die Umgebung sind Arten nicht unveränderlich, sondern *entwickeln* sich im Laufe der Zeit.

  (Die Evolutionstheorie steht damit im Gegensatz zum Kreationismus, der die Unveränderlichkeit der Arten behauptet.)

Prinzipien der organismischen Evolution IV

- **diskrete genetische Einheiten**

  Die Erbinformation wird in diskreten („atomaren“) Einheiten gespeichert, übertragen und geändert (keine kontinuierliche Verschmelzung von Erbmerkmalen), denn sonst kommt es durch Rekombination zum sogenannten *Jenkins nightmare*, dem völligen Verschwinden jeglicher Verschiedenheit in einer Population.

- **Opportunismus**

  Evolutive Prozesse sind äußerst opportunistisch: Sie arbeiten ausschließlich mit dem, was vorhanden ist, nicht mit dem, was es einmal gab oder geben könnte. Bessere oder optimale Lösungen werden nicht gefunden, wenn die erforderlichen evolutiven Zwischenstadien Tauglichkeitsnachteile mit sich bringen.

- **evolutionsstrategische Prinzipien**

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

Prinzipien der organismischen Evolution V

- **ökologische Nischen**

  Konkurrierende Arten können einander tolerieren, wenn sie unterschiedliche Ökonischen („Lebensräume“ im weiten Sinne) besetzen, vielleicht sogar selbst schaffen. Nur so ist — trotz Konkurrenz und natürlicher Auslese — die beobachtete Artenvielfalt möglich.

- **Irreversibilität**

  Der Gang der Evolution ist irreversibel und unwiederholbar.

- **Nichtvorhersagbarkeit**

  Der Gang der Evolution ist nicht determiniert, nicht programmiert, nicht zielgerichtet und deshalb nicht vorhersagbar.

- **wachsende Komplexität**

  Die biologische Evolution hat im allgemeinen zu immer komplexeren Systemen geführt. (Problem: Wie mißt man die Komplexität von Lebewesen?)

Grundlagen genetischer Algorithmen

Grundbegriffe und ihre Bedeutung I

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

Grundbegriffe und ihre Bedeutung II

Begriff	Biologie	Informatik
Phänotyp	äußeres Erscheinungsbild eines Lebewesens	Umsetzung / Implementierung eines Lösungskandidaten
Genotyp	genetische Konstitution eines Lebewesens	Kodierung eines Lösungskandidaten
Population	Menge von Lebewesen	Familie / Multimenge von Chromosomen
Generation	Population zu einem Zeitpunkt	Population zu einem Zeitpunkt
Reproduktion	Erzeugen von Nachkommen aus einem oder mehreren (wenn, dann meist zwei) Lebewesen	Erzeugen von (Kind-)Chromosomen aus einem oder mehreren (Eltern-)Chromosomen
Fitneß	Tauglichkeit / Angepaßtheit eines Lebewesens	Güte / Tauglichkeit eines Lösungskandidaten
	bestimmt Überlebens- und Fortpflanzungschancen	

Elemente eines genetischen Algorithmus I

Ein genetischer Algorithmus besteht aus:

- einer **Kodierungsvorschrift** für die Lösungskandidaten
  Wie die Lösungskandidaten kodiert werden, ist problemspezifisch; es gibt keine allgemeinen Regeln. Wir werden später jedoch einige Aspekte besprechen, die man bei der Wahl einer Kodierung beachten sollte.
- einer Methode, eine **Anfangspopulation** zu erzeugen
  Meist werden einfach zufällige Zeichenketten erzeugt; je nach gewählter Kodierung können aber auch komplexere Verfahren nötig sein.
- einer **Bewertungsfunktion** (Fitneßfunktion) für die Individuen
  Die Bewertungsfunktion spielt die Rolle der Umgebung und gibt die Güte der Individuen an. Meist ist die Bewertungsfunktion mit der zu optimierenden Funktion identisch; sie kann aber auch zusätzliche Elemente enthalten, die einzuhaltende Nebenbedingungen darstellen.
- einer **Auswahlmethode** auf der Grundlage der Fitneßfunktion
  Die Auswahlmethode bestimmt, welche Individuen zur Erzeugung von Nachkommen herangezogen werden oder auch unverändert in die nächste Generation gelangen.

Elemente eines genetischen Algorithmus II

Ein genetischer Algorithmus besteht weiter aus:

- **genetischen Operatoren**, die die Lösungskandidaten ändern
  - *Mutation* — zufällige Veränderung einzelner Gene
  - *Crossover* — Rekombination von Chromosomen

    (eigentlich "crossing over", nach einem Vorgang in der Meiose (Phase der Zellteilung), bei dem Chromsomen zerteilt und überkreuzt wieder zusammengefügt werden)
- Werten für verschiedene **Parameter**

  (z.B. Populationsgröße, Mutationswahrscheinlichkeit etc.)
- einem **Abbruchkriterium**, z.B.
  - eine festgelegte Anzahl von Generationen wurde berechnet
  - eine festgelegte Anzahl von Generationen lang gab es keine Verbesserung
  - eine vorgegebene Mindestlösungsgüte wurde erreicht

Grundstruktur eines genetischen Algorithmus

```
procedure evolution_program;
begin
  t ← 0;                                  (* initialisiere den Generationenzähler *)
  initialize pop(t);                      (* erzeuge die Anfangspopulation *)
  evaluate pop(t);                        (* und bewerte sie (berechne Fitneß) *)
  while not termination criterion do      (* solange Abbruchkriterium nicht erfüllt *)
    t ← t + 1;                            (* zähle die erzeugte Generation *)
    select pop(t) from pop(t − 1);        (* wähle Individuen nach Fitneß aus *)
    alter pop(t);                         (* wende genetische Operatoren an *)
    evaluate pop(t);                      (* bewerte die neue Population *)
  end                                     (* (berechne neue Fitneß) *)
end
```

- Durch die Auswahl wird eine Art „Zwischenpopulation“ von Individuen mit (im Durchschnitt) hoher Fitneß erzeugt.
- Nur die Individuen der Zwischenpopulation können Nachkommen bekommen.

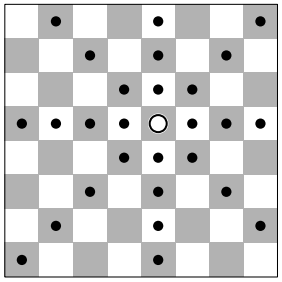
Einführendes Beispiel:

Das n-Damen-Problem

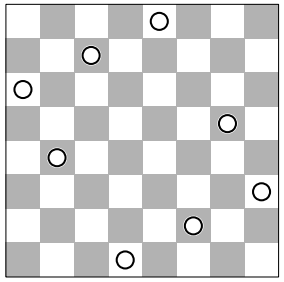
Einführendes Beispiel: Das n-Damen-Problem

Plaziere n Damen auf einem $n \times n$ Schachbrett, so daß in keiner Reihe (Zeile), keiner Linie (Spalte) und keiner Diagonale mehr als eine Dame steht.
Oder: Plaziere die Damen so, daß keine einer anderen im Weg steht.

Zugmöglichkeiten einer Dame



Lösung des 8-Damen-Problems



n-Damen-Problem: Backtracking

- Wie aus anderen Vorlesungen wahrscheinlich bekannt, kann das n-Damen-Problem leicht mit Hilfe des Backtracking gelöst werden:
- Die Damen werden zeilenweise, von unten nach oben, plaziert. (oder auch spaltenweise, von links nach rechts, o.ä.)
- Jede Zeile wird wie folgt bearbeitet:
  - In einer Zeile wird eine Dame der Reihe nach, von links nach rechts, auf die Felder der Zeile gesetzt.
  - Für jede Plazierung der Dame wird geprüft, ob es zu einer Kollision mit Damen in tieferliegenden Zeilen kommt.
  - Ist dies nicht der Fall, wird rekursiv die nächste Zeile bearbeitet.
  - Anschließend wird die Dame ein Feld weiter nach rechts gerückt.
- Kann eine Dame in der obersten Zeile des Brettes plaziert werden, ohne daß es zu einer Kollision kommt, wird die gefundene Lösung ausgegeben.

n-Damen-Problem: Backtracking

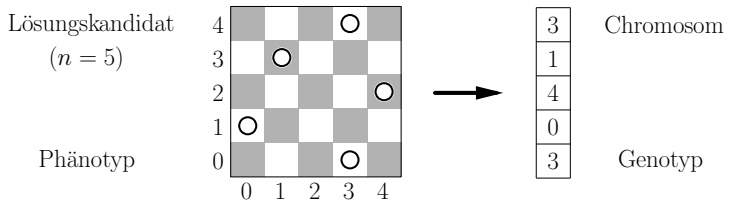
```
int search (int y)
{                                /* --- depth first search */
  int x, i, d;                   /* loop variables, buffer */
  int sol = 0;                   /* solution counter */

  if (y >= size) {               /* if a solution has been found, */
    show(); return 1; }          /* show it and abort the function */
  for (x = 0; x < size; x++) { /* traverse fields of the current row */
    for (i = y; --i >= 0; ) {  /* traverse the preceding rows */
      d = abs(qpos[i] -x);       /* and check for collisions */
      if ((d == 0) || (d == y-i)) break;
    }                            /* if there is a colliding queen, */
    if (i >= 0) continue;        /* skip the current field */
    qpos[y] = x;                 /* otherwise place the queen */
    sol += search(y+1);          /* and search recursively */
  }
  return sol;                    /* return the number of */
}  /* search() */                /* solutions found */
```


n-Damen-Problem: Direkte Berechnung

- Ist man nur an *einer* Lösung (einer Plazierung der Damen) interessiert, so kann man die Positionen für alle $n > 3$ wie folgt berechnen:
- Falls n ungerade ist, setze eine Dame auf $(n-1, n-1)$ und verringere n um 1.
- Falls $n \bmod 6 \neq 2$: Setze die Damen
  in den Zeilen $y = 0, \ldots, \frac{n}{2} - 1$ in die Spalten $x = 2y + 1$,
  in den Zeilen $y = \frac{n}{2}, \ldots, n-1$ in die Spalten $x = 2y - n$.
- Falls $n \bmod 6 = 2$: Setze die Damen
  in den Zeilen $y = 0, \ldots, \frac{n}{2} - 1$ in die Spalten $x = (2y + \frac{n}{2}) \bmod n$,
  in den Zeilen $y = \frac{n}{2}, \ldots, n-1$ in die Spalten $x = (2y - \frac{n}{2} + 2) \bmod n$.
- Es ist daher eigentlich nicht zweckmäßig, einen genetischen Algorithmus zu verwenden, um das n-Damen-Problem zu lösen. Allerdings lassen sich an diesem Problem einige Aspekte genetischer Algorithmen gut verdeutlichen.

Genetischer Algorithmus: Kodierung

- Darstellung eines Lösungskandidaten durch ein Chromosom mit n Genen.
- Jedes Gen beschreibt eine Zeile des Schachbrettes und hat n mögliche Allele. Die Genausprägung gibt die Position der Dame in der zugehörigen Zeile an.



- Man beachte, daß durch die Kodierung bereits Lösungskandidaten mit mehr als einer Dame je Zeile ausgeschlossen werden → kleinerer Suchraum.

Genetischer Algorithmus: Datentypen

- Datentyp für ein Chromosom, der auch die Fitneß speichert.
- Datentyp für eine Population mit einem Puffer für die „Zwischenpopulation“ und einem Merker für das beste Individuum.

```
typedef struct {              /* --- an individual --- */
  int fitness;                /* fitness (number of collisions) */
  int cnt;                    /* number of genes (number of rows) */
  int genes[1];               /* genes (queen positions in rows) */
} IND;                        /* (individual) */

typedef struct {              /* --- a population --- */
  int size;                   /* number of individuals */
  IND **inds;                 /* vector of individuals */
  IND **buf;                  /* buffer for individuals */
  IND *best;                  /* best individual */
} POP;                        /* (population) */
```


Genetischer Algorithmus: Hauptschleife

Die Hauptschleife zeigt die Grundform eines genetischen Algorithmus:

```
pop_init(pop);                /* initialize the population */
while ((pop_eval(pop) < 0)    /* while no solution found and */
&&     (--gencnt >= 0)) {     /* not all generations computed */
  pop_select(pop, tmsize, elitist);
  pop_cross (pop, frac);      /* select individuals, */
  pop_mutate(pop, prob);      /* do crossover, and */
}                             /* mutate individuals */
```

Parameter:

`gencnt`	maximale Anzahl (noch) zu berechnender Generationen
`tmsize`	Größe des Turniers für die Individuenauswahl
`elitist`	zeigt an, ob das beste Individuum immer übernommen werden soll
`frac`	Anteil der Individuen, die Crossover unterworfen werden
`prob`	Mutationswahrscheinlichkeit

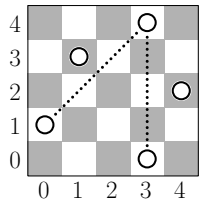
Genetischer Algorithmus: Initialisierung

- Es werden zufällige Folgen von n Zahlen aus $\{0, 1, \ldots, n-1\}$ erzeugt.

```
void ind_init (IND *ind)
{                             /* --- initialize an individual */
  int i;                      /* loop variable */

  for (i = ind->n; --i >= 0; ) /* initialize the genes randomly */
    ind->genes[i] = (int)(ind->n *drand());
  ind->fitness = 1;           /* fitness is not known yet */
}  /* ind_init() */

void pop_init (POP *pop)
{                             /* --- initialize a population */
  int i;                      /* loop variable */

  for (i = pop->size; --i >= 0; )
    ind_init(pop->inds[i]);   /* initialize all individuals */
}  /* pop_init() */
```


Genetischer Algorithmus: Bewertung

- Fitneß: negierte Zahl der Spalten und Diagonalen mit mehr als einer Dame. (Negierte Zahl, damit wir eine zu maximierende Fitneß erhalten.)



2 Kollisionen $\rightarrow$ Fitneß $= -2$

- Bei mehr als zwei Damen in einer Spalte/Diagonale wird jedes Paar gezählt (einfacher zu implementieren).
- Aus dieser Fitneßfunktion ergibt sich unmittelbar das Abbruchkriterium: Eine Lösung hat die (maximal mögliche) Fitneß 0.
- Zusätzlich: Maximale Generationenzahl, um Terminierung zu garantieren.

Genetischer Algorithmus: Bewertung

- Zähle Kollisionen durch Berechnungen auf den Chromosomen.

```
int ind_eval (IND *ind)
{                               /* --- evaluate an individual */
  int i, k;                     /* loop variables */
  int d;                        /* horizontal distance between queens */
  int n;                        /* number of collisions */

  if (ind->fitness <= 0)        /* if the fitness is already known, */
    return ind->fitness;        /* simply return it */
  for (n = 0, i = ind->n; --i > 0; ) {
    for (k = i; --k >= 0; ) {   /* traverse all pairs of queens */
      d = abs(ind->genes[i] -ind->genes[k]);
      if ((d == 0) || (d == i-k)) n++;
    }                           /* count the number of pairs of queens */
  }                             /* in the same column or diagonal */
  return ind->fitness = -n;     /* return the number of collisions */
}  /* ind_eval() */
```


Genetischer Algorithmus: Bewertung

- Es wird die Fitneß aller Individuen der Population berechnet.
- Gleichzeitig wird das beste Individuum bestimmt.
- Hat das beste Individuum die Fitneß 0, so ist eine Lösung gefunden.

```
int pop_eval (POP *pop)
{                               /* --- evaluate a population */
  int i;                        /* loop variable */
  IND *best;                    /* best individual */

  ind_eval(best = pop->inds[0]);
  for (i = pop->size; --i > 0; )
    if (ind_eval(pop->inds[i]) >= best->fitness)
      best = pop->inds[i];      /* find the best individual */
  pop->best = best;             /* note the best individual */
  return best->fitness;         /* and return its fitness */
}  /* pop_eval() */
```


Genetischer Algorithmus: Auswahl von Individuen

Turnierauswahl:

- Es werden `tmsize` zufällig bestimmte Individuen betrachtet.
- Das beste dieser Individuen „gewinnt“ das Turnier und wird ausgewählt.
- Je höher die Fitneß ist, um so größer ist die Chance, ausgewählt zu werden.

```
IND* pop_tmsel (POP *pop, int tmsize)
{                               /* --- tournament selection */
  IND *ind, *best;              /* competing/best individual */

  best = pop->inds[(int)(pop->size *drand())];
  while (--tmsize > 0) {        /* randomly select tmsize individuals */
    ind = pop->inds[(int)(pop->size *drand())];
    if (ind->fitness > best->fitness) best = ind;
  }                             /* det. individual with best fitness */
  return best;                  /* and return this individual */
}  /* pop_tmsel() */
```

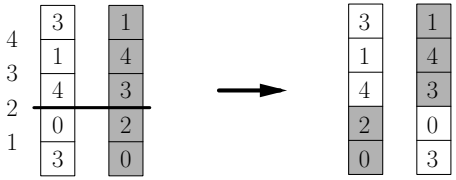

Genetischer Algorithmus: Auswahl von Individuen

- Die Individuen, aus der die Individuen der nächsten Generation erzeugt werden, werden durch Turnierauswahl bestimmt.
- Eventuell wird das beste Individuum übernommen (und nicht verändert).

```
void pop_select (POP *pop, int tmsize, int elitist)
{                               /* --- select individuals */
  int i;                        /* loop variables */
  IND **p;                      /* exchange buffer */

  i = pop->size;                /* select 'popsize' individuals */
  if (elitist)                  /* preserve the best individual */
    ind_copy(pop->buf[--i], pop->best);
  while (--i >= 0)              /* select (other) individuals */
    ind_copy(pop->buf[i], pop_tmsel(pop, tmsize));
  p = pop->inds; pop->inds = pop->buf;
  pop->buf  = p;                /* set selected individuals */
  pop->best = NULL;             /* best individual is not known yet */
}  /* pop_select() */
```

Genetischer Algorithmus: Crossover

- Austausch eines Chromosomenstücks (oder auch einer in anderer Weise ausgewählter Teilmenge der Gene) zwischen zwei Individuen.
- hier: sogenanntes **Ein-Punkt-Crossover**
  - Wähle zufällig eine Trennstelle zwischen zwei Genen.
  - Tausche die Gensequenzen auf der einen Seite dieser Trennstelle aus.
  - Beispiel: Wähle Trennstelle 2.



Fitneß: -2 -3 0 -2

Genetischer Algorithmus: Crossover

- Austausch eines Chromosomenstücks zwischen zwei Individuen.

```
void ind_cross (IND *ind1, IND *ind2)
{                               /* --- crossover of two chromosomes */
  int i;                        /* loop variable */
  int k;                        /* gene index of crossover point */
  int t;                        /* exchange buffer */

  k = (int)(drand() *(ind1->n-1)) +1; /* choose a crossover point */
  if (k > (ind1->n >> 1)) { i = ind1->n; }
  else                    { i = k; k = 0; }
  while (--i >= k) {            /* traverse smaller section */
    t              = ind1->genes[i];
    ind1->genes[i] = ind2->genes[i];
    ind2->genes[i] = t;         /* exchange genes */
  }                             /* of the chromosomes */
  ind1->fitness = 1;            /* invalidate the fitness */
  ind2->fitness = 1;            /* of the changed individuals */
}  /* ind_cross() */
```

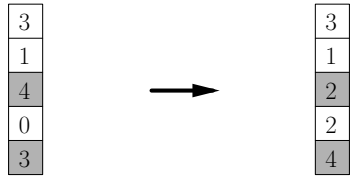

Genetischer Algorithmus: Crossover

- Ein gewisser Anteil der Individuen wird Crossover unterworfen.
- Beide Crossoverprodukte werden in die neue Population aufgenommen, die „Elternindividuen“ gehen verloren.
- Das beste Individuum (falls übernommen) wird keinem Crossover unterzogen.

```
void pop_cross (POP *pop, double frac)
{                               /* --- crossover in a population */
  int i, k;                     /* loop variables */

  k = (int)((pop->size -1) *frac) & ~1;
  for (i = 0; i < k; i += 2)    /* crossover of pairs of individuals */
    ind_cross(pop->inds[i], pop->inds[i+1]);
}  /* pop_cross() */
```


Genetischer Algorithmus: Mutation

- Zufällig gewählte Gene werden zufällig ersetzt (Allele werden geändert).
- Wie viele Gene ersetzt werden, kann auch zufällig gewählt werden. (Die Zahl ersetzter Gene sollte allerdings klein sein.)



Fitneß: -2 -4

- Die meisten Mutationen sind schädlich (verschlechtern die Fitneß).
- Anfangs nicht vorhandene Allele können (nur) durch Mutationen entstehen.

Genetischer Algorithmus: Mutation

- Es wird entschieden, ob eine (weitere) Mutation durchgeführt werden soll.
- Das beste Individuum (falls übernommen) wird nicht mutiert.

```
void ind_mutate (IND *ind, double prob)
{                              /* --- mutate an individual */
  if (drand() >= prob) return; /* det. whether to change individual */
  do ind->genes[(int)(ind->n *drand())] = (int)(ind->n *drand());
  while (drand() < prob);      /* randomly change random genes */
  ind->fitness = 1;            /* fitness is no longer known */
}  /* ind_mutate() */

void pop_mutate (POP *pop, double prob)
{                              /* --- mutate a population */
  int i;                       /* loop variable */
  for (i = pop->size -1; --i >= 0; )
    ind_mutate(pop->inds[i], prob);
}  /* pop_mutate() */          /* mutate individuals */
```


n-Damen-Problem: Programme

- Die besprochenen Verfahren zur Lösung des n-Damen-Problems,
  - Backtracking,
  - direkte Berechnung,
  - genetischer Algorithmus,

  stehen auf der Vorlesungsseite als Kommadozeilenprogramme zur Verfügung. (C-Programme `queens.c` und `qga.c`)
- Ruft man diese Programme ohne Parameter auf, erhält man eine Liste von Programmoptionen.
- Man beachte, daß mit Hilfe des genetischen Algorithmus nicht immer eine Lösung gefunden wird (ausgegebener Lösungskandidat hat eine Fitneß < 0).
- Die Eigenschaften der genannten Verfahren werden anhand dieser Programme in einigen Übungsaufgaben noch genauer betrachtet.

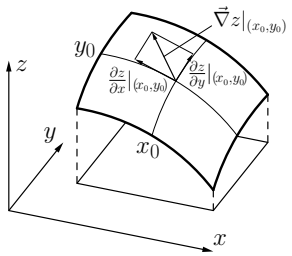
Verwandte Optimierungsverfahren

Verwandte Optimierungsverfahren

- **Allgemeine Problemstellung:**
  Gegeben sei eine Funktion $f : S \to \mathbb{R}$ (S : Suchraum).
  Finde ein Element $s \in S$, das f optimiert (maximiert oder minimiert).
- o.B.d.A.: Finde ein Element $s \in S$, das f maximiert.
  (Ist f zu minimieren, kann man stattdessen $f' \equiv -f$ betrachten.)
- **Voraussetzung:**
  Für ähnliche Elemente $s_1, s_2 \in S$ unterscheiden sich die Funktionswerte $f(s_1)$ und $f(s_2)$ nicht zu sehr (keine großen Sprünge in den Funktionswerten).
- **Verfahren:**
  - Gradientenverfahren
  - Zufallsaufstieg
  - Simuliertes Ausglühen
  - Akzeptieren mit Schwellenwert
  - Sintflut-Algorithmus

Gradientenverfahren

- **Voraussetzung:** $S \subseteq \mathbb{R}^n$, $f : S \to \mathbb{R}$ ist differenzierbar
- **Gradient:** Differentialoperation, die ein Vektorfeld erzeugt.
  Liefert Vektor in Richtung der stärksten Steigung einer Funktion.



- Illustration des Gradienten einer Funktion $z = f(x, y)$ am Punkt (x_0, y_0).
  Es ist $\vec{\nabla} z|_{(x_0,y_0)} = \left(\frac{\partial z}{\partial x}|_{(x_0,y_0)}, \frac{\partial z}{\partial y}|_{(x_0,y_0)}\right)$.

Gradientenverfahren

Idee: Mache, ausgehend von einem zufälligen Startpunkt, kleine Schritte im Suchraum, jeweils in Richtung des stärksten Anstiegs der Funktion, bis ein (lokales) Maximum erreicht ist.

1. Wähle einen (zufälligen) Startpunkt $\vec{x}^{(0)} = \left(x_1^{(0)}, \ldots, x_n^{(0)}\right)$
2. Bestimme den Gradienten am aktuellen Punkt $\vec{x}^{(i)}$:
$$\nabla_{\vec{x}} f\left(\vec{x}^{(i)}\right) = \left(\frac{\partial}{\partial x_1} f\left(\vec{x}^{(i)}\right), \ldots, \frac{\partial}{\partial x_n} f\left(\vec{x}^{(i)}\right)\right)$$
3. Gehe ein kleines Stück in Richtung des Gradienten:
$$\vec{x}^{(i+1)} = \vec{x}^{(i)} + \eta \, \nabla_{\vec{x}} f\left(\vec{x}^{(i)}\right).$$
   η ist ein Schrittweitenparameter („Lernrate“ in neuronalen Netzen)
4. Wiederhole Schritte 2 und 3, bis ein Abbruchkriterium erfüllt ist.
   (Vorgegebene Anzahl Schritte ausgeführt, aktueller Gradient sehr klein)

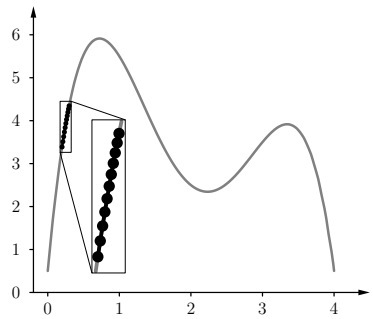
Gradientenverfahren: Probleme

- **Wahl des Schrittweitenparameters**
  - Bei einem zu kleinen Wert kann es sehr lange dauern, bis das Maximum erreicht ist (Schritte zu klein).
  - Bei einem zu großen Wert kann es zu Oszillationen (Hin- und Herspringen im Suchraum) kommen (Schritte zu groß).
  - Lösungsmöglichkeiten: Momentterm, adaptiver Schrittweitenparameter (Details: siehe Vorlesung „Neuronale Netze“)
- **Hängenbleiben in lokalen Maxima**
  - Da nur lokale Steigungsinformation genutzt wird, wird eventuell nur ein lokales Maximum erreicht.
  - Dieses Problem kann *nicht* prinzipiell behoben werden.
  - Chancenverbesserung für Finden des globalen Optimums: Mehrfaches Ausführen des Gradientenaufstieg von verschiedenen Startwerten aus.

Gradientenverfahren: Beispiele

Beispielfunktion: $$f(x) = -\frac{5}{6}x^4 + 7x^3 - \frac{115}{6}x^2 + 18x + \frac{1}{2},$$

i	x_i	$f(x_i)$	$f'(x_i)$	Δx_i
0	0.200	3.388	11.147	0.011
1	0.211	3.510	10.811	0.011
2	0.222	3.626	10.490	0.010
3	0.232	3.734	10.182	0.010
4	0.243	3.836	9.888	0.010
5	0.253	3.932	9.606	0.010
6	0.262	4.023	9.335	0.009
7	0.271	4.109	9.075	0.009
8	0.281	4.191	8.825	0.009
9	0.289	4.267	8.585	0.009
10	0.298	4.340		

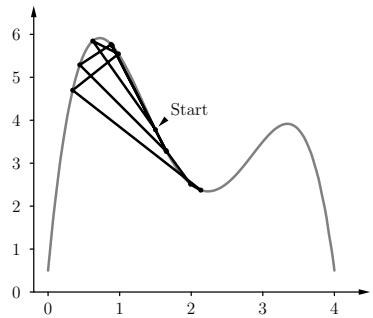


Gradientenaufstieg mit Startwert 0.2 und Schrittweitenparameter 0.001.

Gradientenverfahren: Beispiele

Beispielfunktion: $$f(x) = -\frac{5}{6}x^4 + 7x^3 - \frac{115}{6}x^2 + 18x + \frac{1}{2},$$

i	x_i	$f(x_i)$	$f'(x_i)$	Δx_i
0	1.500	3.781	−3.500	−0.875
1	0.625	5.845	1.431	0.358
2	0.983	5.545	−2.554	−0.639
3	0.344	4.699	7.157	1.789
4	2.134	2.373	−0.567	−0.142
5	1.992	2.511	−1.380	−0.345
6	1.647	3.297	−3.063	−0.766
7	0.881	5.766	−1.753	−0.438
8	0.443	5.289	4.851	1.213
9	1.656	3.269	−3.029	−0.757
10	0.898	5.734		

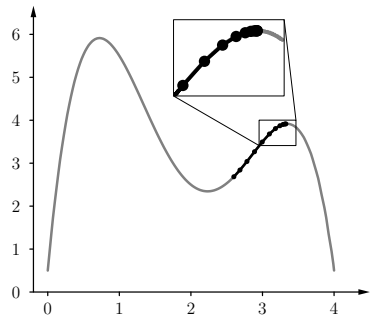


Gradientenaufstieg mit Startwert 1.5 und Schrittweitenparameter 0.25.

Gradientenverfahren: Beispiele

Beispielfunktion: $$f(x) = -\frac{5}{6}x^4 + 7x^3 - \frac{115}{6}x^2 + 18x + \frac{1}{2},$$

i	x_i	$f(x_i)$	$f'(x_i)$	Δx_i
0	2.600	2.684	1.707	0.085
1	2.685	2.840	1.947	0.097
2	2.783	3.039	2.116	0.106
3	2.888	3.267	2.153	0.108
4	2.996	3.492	2.009	0.100
5	3.097	3.680	1.688	0.084
6	3.181	3.805	1.263	0.063
7	3.244	3.872	0.845	0.042
8	3.286	3.901	0.515	0.026
9	3.312	3.911	0.293	0.015
10	3.327	3.915		



Gradientenaufstieg mit Startwert 2.6 und Schrittweitenparameter 0.05.

Zufallsaufstieg

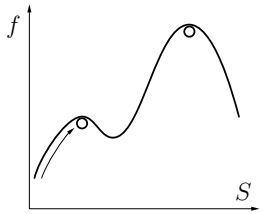
- **Idee:** Wenn die Funktion f nicht differenzierbar ist, kann man versuchen, eine Richtung, in der die Funktion f ansteigt, durch Auswerten zufälliger Punkte in der Umgebung des aktuellen Punktes zu bestimmen.

1. Wähle einen (zufälligen) Startpunkt $s_0 \in S$.
2. Wähle einen Punkt $s' \in S$ „in der Nähe“ des aktuellen Punktes s_i. (z.B. durch zufällige, aber nicht zu große Veränderung von s_i)
3. Setze
$$s_{i+1} = \begin{cases} s', & \text{falls } f(s') \geq f(s_i), \\ s_i, & \text{sonst.} \end{cases}$$
4. Wiederhole Schritte 2 und 3, bis ein Abbruchkriterium erfüllt ist.

- **Problem:** Hängenbleiben in lokalen Maxima.
  Alle folgenden Verfahren versuchen, dieses Problem zu verringern.

Simuliertes Ausglühen

- Kann als Erweiterung des Zufalls- und Gradientenaufstiegs gesehen werden, die ein Hängenbleiben vermeidet.
- **Idee:** Übergänge von niedrigeren zu höheren (lokalen) Maxima sollen wahrscheinlicher sein als umgekehrt.



Prinzip des simulierten Ausglühens:

- Zufällige Varianten der aktuellen Lösung werden erzeugt.
- Bessere Lösungen werden immer übernommen.
- Schlechtere Lösungen werden mit einer bestimmten Wahrscheinlichkeit übernommen, die abhängt von
  - der Qualitätsdifferenz der aktuellen und der neuen Lösung und
  - einem Temperaturparameter, der im Laufe der Zeit verringert wird.

Simuliertes Ausglühen

- **Motivation:** (Minimierung statt Maximierung)

  Physikalische Minimierung der Energie (genauer: der Atomgitterenergie), wenn ein erhitztes Stück Metall langsam abgekühlt wird.

  Dieser Prozeß wird **Ausglühen** (engl.: annealing) genannt. Er dient dazu, ein Metall weicher zu machen, indem innere Spannungen und Instabilitäten aufgehoben werden, um es dann leichter bearbeiten zu können.

- **Alternative Motivation:** (ebenfalls Minimierung)

  Eine Kugel rollt auf einer unregelmäßig gewellten Oberfläche.
  Die zu minimierende Funktion ist die potentielle Energie der Kugel.

  Am Anfang hat die Kugel eine gewisse kinetische Energie, die es ihr erlaubt, Anstiege zu überwinden. Durch Reibung sinkt die Energie der Kugel, so daß sie schließlich in einem Tal zur Ruhe kommt.

- **Achtung:** Es gibt keine Garantie, daß das globale Optimum gefunden wird.

Simuliertes Ausglühen

1. Wähle einen (zufälligen) Startpunkt $s_0 \in S$.
2. Wähle einen Punkt $s' \in S$ „in der Nähe“ des aktuellen Punktes s_i.
   (z.B. durch zufällige, aber nicht zu große Veränderung von s_i)
3. Setze
   $$s_{i+1} = \begin{cases} s', & \text{falls } f(s') \geq f(s_i), \\ s' \quad \text{mit Wahrscheinlichkeit } p = e^{-\frac{\Delta f}{kT}}, & \text{sonst.} \\ s_i \quad \text{mit Wahrscheinlichkeit } 1-p, & \end{cases}$$

$\Delta f = f(s_i) - f(s')$	Qualitätsverringerung der Lösung
$k = \Delta f_{\max}$	(Schätzung der) Spannweite der Funktionswerte
T	Temperaturparameter; wird im Laufe der Zeit gesenkt

4. Wiederhole Schritte 2 und 3, bis ein Abbruchkriterium erfüllt ist.

- Für kleine T geht das Verfahren nahezu in einen Zufallsaufstieg über.

Akzeptieren mit Schwellenwert

- **Idee:** Ähnlich wie beim simulierten Ausglühen werden auch schlechtere Lösungen akzeptiert, allerdings mit einer oberen Schranke für die Verschlechterung.

1. Wähle einen (zufälligen) Startpunkt $s_0 \in S$.
2. Wähle einen Punkt $s' \in S$ „in der Nähe“ des aktuellen Punktes s_i.
   (z.B. durch zufällige, aber nicht zu große Veränderung von s_i)
3. Setze
   $$s_{i+1} = \begin{cases} s', & \text{falls } f(s') \geq f(s_i) - \theta, \\ s_i, & \text{sonst.} \end{cases}$$

   θ Schwellenwert für das Akzeptieren schlechterer Lösungen;
   wird im Laufe der Zeit (langsam) gesenkt.
   ($\theta = 0 \quad \rightarrow \quad$ normaler Zufallsaufstieg)
4. Wiederhole Schritte 2 und 3, bis ein Abbruchkriterium erfüllt ist.

Sintflut-Algorithmus

- **Idee:** Ähnlich wie beim simulierten Ausglühen werden auch schlechtere Lösungen akzeptiert, allerdings mit einer unteren Schranke.

1. Wähle einen (zufälligen) Startpunkt $s_0 \in S$.
2. Wähle einen Punkt $s' \in S$ „in der Nähe“ des aktuellen Punktes s_i.
   (z.B. durch zufällige, aber nicht zu große Veränderung von s_i)
3. Setze
   $$s_{i+1} = \begin{cases} s', & \text{falls } f(s') \geq \theta, \\ s_i, & \text{sonst.} \end{cases}$$

   θ Untere Schranke für die Lösungsgüte;
   wird im Laufe der Zeit (langsam) erhöht.
   („Fluten“ des Funktionsgebirges, θ entspricht dem „Wasserstand“)
4. Wiederhole Schritte 2 und 3, bis ein Abbruchkriterium erfüllt ist.

Beispiel: Problem des Handlungsreisenden

- **Gegeben:**
  - Eine Menge von n Städten (als Punkte in einer Ebene)
  - Abstände/Kosten der Wege zwischen den Städten
- **Gesucht:**
  - Rundreise minimaler Länge/Kosten durch alle n Städte, auf der keine Stadt mehr als einmal besucht wird
- **Mathematisch:** Suche eines Hamiltonkreises (enthält jeden Knoten genau einmal) mit minimalem Gewicht in einem Graphen mit gewichteten Kanten.
- **Bekannt:** Dieses Problem ist NP-vollständig, d.h., man kennt keinen Algorithmus, der dieses Problem in polynomialer Zeit löst.
- **Daher:** Für großes n ist in annehmbarer Zeit nur eine Näherungslösung berechenbar (die beste Lösung kann gefunden werden, dies ist aber nicht garantiert).
- **Hier:** Betrachte Ansatz mit simuliertem Ausglühen und Zufallsaufstieg.

Beispiel: Problem des Handlungsreisenden

1. Bringe die Städte in eine zufällige Reihenfolge (zufällige Rundreise).
2. Wähle zufällig zweimal zwei Städte, die in der aktuellen Rundreise aufeinander folgen (alle vier Städte verschieden). Trenne die Rundreise zwischen den Städten jedes Paares auf und drehe den dazwischenliegenden Teil um.
3. Wenn die so entstehende Rundreise besser (kürzer, billiger) ist als die alte, ersetze die alte Rundreise auf jeden Fall durch die neue, sonst ersetze die alte Rundreise nur mit Wahrscheinlichkeit $p = e^{-\frac{\Delta Q}{kT}}$.

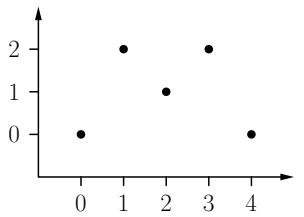
   ΔQ Qualitätsunterschied zwischen alter und neuer Rundreise

   k Spannweite der Rundreisequalitäten (muß ggf. geschätzt werden, z.B. $k_i = \frac{i+1}{i} \max_{j=1}^{i} \Delta Q_j$, wobei ΔQ_j der Qualitätsunterschied im j-ten Schritt und i der aktuelle Schritt ist.)

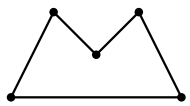
   T Temperaturparameter, der im Laufe der Zeit abnimmt, z.B. $T = \frac{1}{i}$.
4. Wiederhole die Schritte 2 und 3, bis ein Abbruchkriterium erfüllt ist.

Beispiel: Problem des Handlungsreisenden

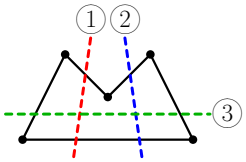
- Reiner Zufallsaufstieg kann in einem lokalen Minimum hängenbleiben. Dazu ein einfaches Beispiel mit 5 Städten:



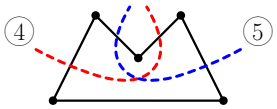
anfängliche Rundreise



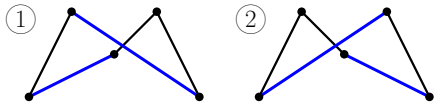
Länge: $2\sqrt{2} + 2\sqrt{5} + 4 \approx 11.30$



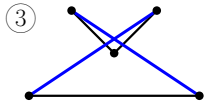
mögliche Teilungen der Rundreise



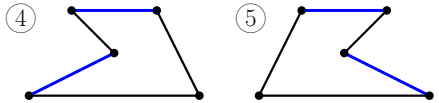
Beispiel: Problem des Handlungsreisenden



Länge: $\sqrt{2} + 3\sqrt{5} + \sqrt{13} \approx 11.73$

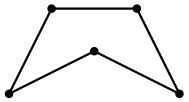


Länge: $\sqrt{2} + 2\sqrt{13} + 4 \approx 14.04$



Länge: $\sqrt{2} + 2\sqrt{5} + 2 + 4 \approx 11.89$

Beste Rundreise:
(globales Optimum)



Länge: $4\sqrt{5} + 2 \approx 10.94$

Beispiel: Problem des Handlungsreisenden

- Alle Modifikationen der Anfangsrundreise führen zu Rundreisen, die schlechter sind. Das globale Optimum kann daher, ausgehend von dieser Rundreise, mit einem reinen Zufallsaufstieg nicht gefunden werden.
- Beim simulierten Ausglühen werden dagegen mitunter auch schlechtere Lösungen akzeptiert, so daß man zum globalen Optimum gelangen kann.
  (*aber*: Es gibt keine Garantie, daß dies geschieht!)
- **Beachte:** Es kann von den erlaubten Operationen abhängen, ob die Suche in einem lokalen Optimum hängenbleiben kann:

  Läßt man als weitere Operation zu, daß die Position einer Stadt in der Rundreise geändert wird (Entfernen von der aktuellen Position und Einfügen an einer anderen), so tritt im betrachteten Beispiel kein Hängenbleiben mehr auf.

  Auch für diese Operationenmenge läßt sich jedoch ein Beispiel konstruieren, in dem die Suche in einem lokalen Minimum hängenbleibt.

Verwandte Optimierungsverfahren: Probleme

- Alle bisher betrachteten Verfahren suchen i.w. **lokal**:
  - Es wird stets nur ein aktueller Lösungskandidat betrachtet.
  - Der aktuelle Lösungskandidat wird nur geringfügig verändert.
- **Nachteil:** Es wird u.U. nur ein kleiner Teil des Suchraums betrachtet.
- **Abhilfe:** Mehrere Läufe des Verfahrens mit verschiedenen Startpunkten.
  Nachteil: Keine Informationsübertragung von einem Lauf zum nächsten.
- Beachte: Große Veränderungen des Lösungskandidaten, bis hin zu völliger Neuberechnung, sind nicht sinnvoll, da dann zu wenig/keine Information von einem Lösungskandidaten zum nächsten weitergegeben wird.

  (Man könnte im Extremfall auch einfach eine Menge von Lösungskandidaten zufällig erzeugen und den besten auswählen.)
- $\rightarrow$ wichtig sind: Zusammenhang der Lösungskandidaten,
  großräumigere Abdeckung des Suchraums

Teilchenschwarmoptimierung

(engl.: **Particle Swarm Optimization**) [Kennedy and Eberhart 1995]

- Teilchenschwarmoptimierung kann gesehen werden als ein Verfahren, das Elemente der bahnorientierten Suche (z.B. Gradientenverfahren) und populationsbasierter Suche (z.B. genetische Algorithmen) zusammenbringt.
- **Ansatz:** Statt nur einem einzelnen aktuellen Lösungskandidaten wird ein „Schwarm“ von m Lösungskandidaten verwendet.
- **Motivation:** Verhalten von z.B. Fischschwärmen bei der Futtersuche:
  Zufälliges Ausschwärmen, aber stets auch Rückkehr zum Schwarm.
  Informationsaustausch zwischen den Schwarmmitgliedern.
- **Voraussetzung:** Der Suchraum ist reellwertig, d.h. $S \subseteq \mathbb{R}^n$ und folglich die zu optimierende (o.B.d.A.: zu maximierende) Funktion $f : \mathbb{R}^n \rightarrow \mathbb{R}$.
- **Vorgehen:** Jeder Lösungskandidat wird als „Teilchen“ aufgefaßt, das einen Ort $\vec{x}_i$ im Suchraum und eine Geschwindigkeit $\vec{v}_i$ hat, $i = 1, \ldots, m$.

Teilchenschwarmoptimierung

- **Aktualisierungsformeln** für Ort und Geschwindigkeit des i-ten Teilchens:

$$\begin{aligned}\vec{x}_i(t+1) &= \vec{x}_i(t) + \vec{v}_i(t)\\ \vec{v}_i(t+1) &= \alpha \cdot \vec{v}_i(t) + \beta_1 \cdot \Big(\vec{x}_i^{\,(\mathrm{local})}(t) - \vec{x}_i(t)\Big) + \beta_2 \cdot \Big(\vec{x}^{\,(\mathrm{global})}(t) - \vec{x}_i(t)\Big)\end{aligned}$$

- $\vec{x}_i^{\,(\mathrm{local})}$ ist das **lokale Gedächtnis** des Individuums (Teilchens).
  Es ist der beste Ort im Suchraum, den das Teilchen bisher besucht hat, d.h.

$$\vec{x}_i^{\,(\mathrm{local})}(t) = \vec{x}_i\left(\mathrm{argmax}_{u=1}^{t} f(\vec{x}_i(u))\right)$$

- $\vec{x}^{\,(\mathrm{global})}$ ist das **globale Gedächtnis** des Schwarms.
  Es ist der beste Ort im Suchraum, den ein Individuum des Schwarms bisher besucht hat (bester bisher gefundener Lösungskandidat), d.h.

$$\vec{x}^{\,(\mathrm{global})}(t) = \vec{x}_j^{\,(\mathrm{local})}(t) \qquad \text{mit} \qquad j = \mathrm{argmax}_{i=1}^{m}\, f\Big(\vec{x}_i^{\,(\mathrm{local})}(t)\Big).$$

- Parameterwahl: β_1, β_2 zufällig in jedem Schritt, α mit der Zeit abnehmend.

Formen genetischer Algorithmen

Grundstruktur eines genetischen Algorithmus

```
procedure evolution_program;
begin
  t ← 0;                                  (* initialisiere den Generationenzähler *)
  initialize pop(t);                      (* erzeuge die Anfangspopulation *)
  evaluate pop(t);                        (* und bewerte sie (berechne Fitneß) *)
  while not termination criterion do      (* solange Abbruchkriterium nicht erfüllt *)
    t ← t + 1;                            (* zähle die erzeugte Generation *)
    select pop(t) from pop(t − 1);        (* wähle Individuen nach Fitneß aus *)
    alter pop(t);                         (* wende genetische Operatoren an *)
    evaluate pop(t);                      (* bewerte die neue Population *)
  end                                     (* (berechne neue Fitneß) *)
end
```

- Durch die Auswahl wird eine Art „Zwischenpopulation“ von Individuen mit (im Durchschnitt) hoher Fitneß erzeugt.
- Nur die Individuen der Zwischenpopulation können Nachkommen bekommen.

Genetische Algorithmen: Formen

- **Standardform:**
  - Auswahl einer „Zwischenpopulation“ mit einem Selektionsverfahren
  - Anwendung der genetischen Operatoren auf die Zwischenpopulation, wobei alle Individuen mutiert werden können.
- **Modifizierte Form:** (Parameter r, $0 < r <$ popsize, und p_c, $0 < p_c < 1$)
  - Wähle aus pop(t) zufällig (aber unter Berücksichtigung der Fitneß) r Chromosomen (**mit** Zurücklegen).
  - Bestimme für jedes dieser r Chromosomen, ob es am Crossover teilnimmt (Wahrscheinlichkeit p_c) oder mutiert wird (Wahrscheinlichkeit $1 - p_c$) und wende die genetischen Operatoren an.
  - Wähle aus pop(t) zufällig (aber unter Berücksichtigung der Fitneß) popsize $-r$ Chromosomen (**ohne** Zurücklegen).
  - Diese popsize $-r$ Chromosomen werden unverändert übernommen.

Genetische Algorithmen: Formen

Eigenschaften der modifizierten Form:

- Ein Chromosom nimmt **entweder** am Crossover teil **oder** wird mutiert.
- Die unveränderten Chromosomen werden nicht vervielfacht (da sie *ohne* Zurücklegen gewählt werden).

Vorteile:

- Die Population enthält so gut wie keine identischen Individuen. (Prinzip der *Diversität* — Verschiedenheit der Individuen)
- Vorteilhafte Crossoverprodukte haben bessere Chancen, da sie nicht durch eventuelle Mutationen beeinträchtigt werden.

Nachteile:

- Größere Gefahr des „Aussterbens“ guter Lösungskandidaten, da es nur eine Kopie jedes Individuums in einer Population gibt.

Genetische Algorithmen: Formen

Steady-State genetischer Algorithmus (Parameter r, $1 \leq r \leq$ popsize)

- Zufällige Auswahl von zwei Eltern (unter Berücksichtigung der Fitneß) und Anwendung genetischer Operatoren auf diese Eltern.
- Verwerfen von Duplikaten (in der Population bereits vorhandene Individuen).
- Insgesamt werden so r Nachkommen erzeugt, die die r schlechtesten Nachkommen der Population ersetzen ($1 \leq r \leq$ popsize, oft $r = 2$).

Vorteile:

- Duplikate werden explizit vermieden $\rightarrow$ bessere Abdeckung des Suchraums
- kein Aussterben sehr guter Individuen (wenn $q <$ popsize)

Nachteile:

- Größerer Aufwand bei der Individuenauswahl.
- Etwas größere Gefahr des Hängenbleibens in lokalen Optima.

Elemente genetischer Algorithmen

1. Kodierung der Lösungskandidaten

Genetische Algorithmen: Kodierung

Im folgenden betrachten wir die Elemente eines genetischen Algorithmus genauer.

Zunächst: **Kodierung der Lösungskandidaten**

- Wie bereits erwähnt, ist die Kodierung problemspezifisch zu wählen.
- Es gibt kein allgemeines „Kochrezept“, um eine (gute) Kodierung zu finden.
- Aber man kann einige Prinzipien angeben, die beachtet werden sollten.

Wünschenswerte Eigenschaften einer Kodierung:

- Ähnliche Phänotypen sollten durch ähnliche Genotypen dargestellt werden.
- Ähnlich kodierte Lösungskandidaten sollten eine ähnliche Fitneß haben.
- Der Suchraum (die Menge möglicher Lösungskandidaten) sollte, soweit möglich, unter den verwendeten genetischen Operatoren abgeschlossen sein.

Genetische Algorithmen: Kodierung

Ähnliche Phänotypen sollten durch ähnliche Genotypen dargestellt werden.

- Mutationen einzelner Gene führen zu ähnlichen Genotypen (einzelne Alleländerungen $\rightarrow$ kleine Änderung des Chromosoms).
- Werden ähnliche Phänotypen nicht durch ähnliche Genotypen dargestellt, können naheliegende Verbesserungen u.U. nicht erzeugt werden.

  (Es ist dann eine große Änderung des Genotyps erforderlich, um zu einem ähnlichen (und vielleicht besseren) Phänotyp zu gelangen.)

Beispiel zur Verdeutlichung:

- Optimierung einer reellen Funktion $y = f(x_1, \ldots, x_n)$.
- Die (reellen) Argumente sollen durch Binärcodes dargestellt werden.
- Problem: einfache Kodierung als Binärzahl führt zu „Hamming-Klippen“.

Binärkodierung reeller Zahlen

- **Gegeben:** ein reelles Interval $[a, b]$ und eine Kodierungsgenauigkeit ε.
- **Gesucht:** eine Kodierungsvorschrift für Zahlen $x \in [a, b]$ als Binärzahl z, so daß die kodierte Zahl um weniger als ε von ihrem tatsächlichen Wert abweicht.
- **Vorgehen:**
  - Teile das Intervall $[a, b]$ in gleich große Abschnitte mit einer Länge $\leq \varepsilon$.
    $\rightarrow 2^k$ Abschnitte mit $k = \left\lceil \log_2 \frac{b-a}{\varepsilon} \right\rceil$ kodiert durch Zahlen $0, \ldots, 2^k - 1$.
  - *Kodierung:* $z = \left\lfloor \frac{x-a}{b-a}(2^k - 1) \right\rfloor$ $\left(\text{alternativ: } \left\lfloor \frac{x-a}{b-a}(2^k - 1) + \frac{1}{2} \right\rfloor, \right.$
  - *Dekodierung:* $x = a + z \cdot \frac{b-a}{2^k - 1}$ $\left. \text{dann reicht } k = \left\lceil \log_2 \frac{b-a}{2\varepsilon} \right\rceil \right)$
- **Beispiel:** Intervall $[-1, 2]$, Genauigkeit $\varepsilon = 10^{-6}$, $x = 0.637197$.
  - $k = \left\lceil \log_2 \frac{2-(-1)}{10^{-6}} \right\rceil = \left\lceil \log_2 3 \cdot 10^6 \right\rceil = 22$
  - $z = \left\lfloor \frac{0.637197-(-1)}{2-(-1)}(2^{22} - 1) \right\rfloor = 2288966_{10} = 1000101110110101000110_2$

Vermeidung von Hamming-Klippen: Gray-Kodes

Problem: Benachbarte Zahlen können sehr verschieden kodiert sein, d.h. die Kodierungen haben einen großen Hamming-Abstand (Anzahl verschiedener Bits). Große Hamming-Abstände können durch Mutationen und Crossover nur sehr schwer überwunden werden (sogenannte „Hamming-Klippen“).

Beispiel: Der Bereich der Zahlen von 0 bis 1 werde durch 4-Bit-Zahlen dargestellt, d.h. Abbildung $\frac{k}{15} \rightarrow k$. Dann haben die Kodierungen von $\frac{7}{15}$ (0111) und $\frac{8}{15}$ (1000) den Hamming-Abstand 4, denn jedes Bit ist verschieden.

Lösung: Gray-Kodes — benachbarte Zahlen unterscheiden sich nur in einem Bit.

binär	Gray
0000	0000
0001	0001
0010	0011
0011	0010

binär	Gray
0100	0110
0101	0111
0110	0101
0111	0100

binär	Gray
1000	1100
1001	1101
1010	1111
1011	1110

binär	Gray
1100	1010
1101	1011
1110	1001
1111	1000

Gray-Kodes: Berechnung

- **Gray-Kodes sind nicht eindeutig:**
  Jeder Kode, in dem sich die Kodierungen benachbarter Zahlen nur in einem Bit unterscheiden, heißt Gray-Kode.
- Gray-Kodes werden meist aus einer Binärzahlkodierung berechnet.
- **Häufigste Form:**
  - *Kodierung:* $g = z \oplus \left\lfloor \frac{z}{2} \right\rfloor$ ($\oplus$: Exklusiv-Oder der Binärdarstellung)
  - *Dekodierung:* $z = \bigoplus_{i=0}^{k-1} \left\lfloor \frac{g}{2^i} \right\rfloor$
- **Beispiel:** Intervall $[-1, 2]$, Genauigkeit $\varepsilon = 10^{-6}$, $x = 0.637197$.
  - $z = \left\lfloor \frac{0.637197-(-1)}{2-(-1)}(2^{22} - 1) \right\rfloor = 2288966_{10} = 1000101110110101000110_2$
  - $g = 1000101110110101000110_2$
    $\oplus \; 1000101110110101000011_2$
    $= 1100111001101111100101_2$

Gray-Kodes: Implementierung

```
unsigned int num2gray (unsigned int x)
{                                /* --- convert number to Gray code */
  return x ^ (x >> 1);           /* exclusive or with shifted input */
}  /* num2gray() */
```

```
unsigned int gray2num (unsigned int x)
{                                /* --- convert Gray code to number */
  unsigned int y = x;            /* copy the input number and */
  while (x >>= 1) y ^= x;        /* do an exclusive or with all */
  return y;                      /* shift downs of the input */
}  /* gray2num() */              /* and return the result */
```

```
unsigned int gray2num (unsigned int x)
{                                /* --- convert Gray code to number */
  x ^= x >> 16; x ^= x >> 8;     /* do an exclusive or with all */
  x ^= x >>  4; x ^= x >> 2;     /* shift downs of the input */
  return x ^ (x >> 1);           /* and return the result */
}  /* gray2num() */              /* (32 bit integers only) */
```

Genetische Algorithmen: Kodierung

Ähnlich kodierte Lösungskandidaten sollten eine ähnliche Fitneß haben.

- **Problem der Epistasie:**
  - *in der Biologie:* Ein Allel eines Gens — des sogenannten epistatischen Gens — unterdrückt die Wirkung aller möglichen Allele eines anderen Gens (oder auch mehrerer anderer Gene).
  - *in genetischen Algorithmen:*
    Wechselwirkung zwischen den Genen eines Chromosoms.

    Die Änderung der Fitneß durch die Änderung eines Gens hängt stark von den Ausprägungen der anderen Gene ab.
- **Epistasie in der Biologie:**
  - Abweichungen von den Mendelschen Gesetzen beruhen oft auf Epistasie.
  - Kreuzt man reinerbige schwarz- und weißsamige Bohnen, so erhält man in der zweiten Nachkommengeneration schwarz-, weiß- und braunsamige Bohnen im Verhältnis 12:1:3, was den Mendelschen Gesetzen widerspricht.

Genetische Algorithmen: Epistasie

Beispiel zur Epistasie:

- **Problem des Handlungsreisenden**
  (Finde Rundreise mit minimalen Kosten durch n Städte.)
- **Kodierung 1:**
  Eine Rundreise wird durch eine Permutation der Städte dargestellt.
  (Stadt an k-ter Position wird im k-ten Schritt besucht.)

  Geringe Epistasie: z.B. Austausch zweier Städte ändert die Fitneß (Kosten der Rundreise) i.a. etwa gleich stark (lokale Touränderung).
- **Kodierung 2:**
  Die Rundreise wird durch Angabe der Position der jeweils nächsten Stadt in einer Liste beschrieben, aus der alle besuchten Städte gestrichen werden.

  Hohe Epistasie: Die Änderung eines Gens, speziell im Chromosom vorn liegender Gene, kann (fast) die gesamte Rundreise ändern (globale Touränderung) und führt daher oft zu großen Änderungen der Fitneß.

Genetische Algorithmen: Epistasie

Erläuterungen zur Kodierung 2: Wirkung einer Mutation

	Chromosom	Liste noch zu besuchender Städte	Rundreise
vor Mutation	5	1, 2, 3, 4, **5**, 6	5
	3	1, 2, **3**, 4, 6	3
	3	1, 2, **4**, 6	4
	2	1, **2**, 6	2
	2	1, **6**	6
	1	**1**	1

	Chromosom	Liste noch zu besuchender Städte	Rundreise
nach Mutation	**1**	**1**, 2, 3, 4, 5, 6	1
	3	2, 3, **4**, 5, 6	4
	3	2, 3, **5**, 6	5
	2	2, **3**, 6	3
	2	2, **6**	6
	1	**2**	2

Genetische Algorithmen: Epistasie

- Zeigt die verwendete Kodierung hohe Epistasie, so ist das Optimierungsproblem oft für einen genetischen Algorithmus schwer zu lösen, da Regelmäßigkeiten fehlen, die durch ihn ausgenutzt werden könnten.
  (Mutation und Crossover führen zu fast zufälligen Fitneßänderungen.)
- Läßt sich eine Kodierung mit sehr geringer Epistasie finden, so sind andere Verfahren oft besser geeignet (z.B. Zufallsaufstieg).
- Man hat versucht, mit dem Begriff der Epistasie Probleme als durch einen genetischen Algorithmus leicht oder schwer lösbar zu kennzeichnen [Davidor 1990]. Das gelingt jedoch nicht:
  - Epistasie ist eine Eigenschaft der Kodierung, **nicht** des Problems.

    (Es kann für ein Problem Kodierungen mit hoher und niedriger Epistasie geben — siehe vorangehendes Beispiel.)
  - Es gibt Probleme, die mit geringer Epistasie kodiert werden können, und dennoch durch einen genetischen Algorithmus schwer zu lösen sind.

Genetische Algorithmen: Kodierung

Der Suchraum (Menge der kodierten Lösungskandidaten) sollte, soweit möglich, unter den verwendeten genetischen Operatoren abgeschlossen sein.

- Was als Verlassen des Suchraums gilt, ist u.U. eine Definitionsfrage.
- Allgemein: Der **Suchraum wird verlassen**, wenn
  - das neue Chromosom nicht sinnvoll interpretiert/dekodiert werden kann,
  - der Lösungskandidat bestimmte prinzipielle Anforderungen nicht erfüllt,
  - der Lösungskandidat durch die Fitneßfunktion falsch bewertet wird.
- Problem der **Abstimmung** von Kodierung und genetischen Operatoren.
  - Verwenden kodierungsspezifischer genetischer Operatoren.
  - Einsatz von Mechanismen, die Chromosomen „reparieren“.
  - Einführen eines Strafterms, der die Fitneß von Chromosomen außerhalb des Suchraums (deutlich) verringert.

Genetische Algorithmen: Verlassen des Suchraums

Beispiel zum Verlassen des Suchraums:

- n-Damen-Problem (Plazierung von n-Damen auf einem $n \times n$-Schachbrett).
- **Kodierung 1:**
  Chromosom der Länge n, das die Spaltenpositionen der Damen je Zeile angibt (Allele $0, \ldots, n-1$, siehe einführendes Beispiel).
  Operatoren: Ein-Punkt-Crossover, Standardmutation
  Es entstehen stets wieder gültige Vektoren von Spaltenpositionen.
  $\rightarrow$ Suchraum wird nicht verlassen.
- **Kodierung 2:**
  Chromosom der Länge n, das die Nummern der Felder (Allele $0, \ldots, n^2-1$) angibt, auf denen Damen stehen.
  Operatoren: Ein-Punkt-Crossover, Standardmutation
  Es entstehen Chromosomen, die mehrere Damen auf das gleiche Feld setzen.
  $\rightarrow$ Suchraum wird verlassen.

Genetische Algorithmen: Verlassen des Suchraums

Mögliche Lösungsansätze am Beispiel des n-Damen-Problems:

- **Andere Kodierung verwenden:** Da die erste Kodierung das Problem des Verlassens des Suchraums vermeidet, außerdem der Suchraum deutlich kleiner ist, ist sie vorzuziehen. (Dies ist, wenn durchführbar, die beste Variante!)
- **Kodierungsspezifische genetische Operatoren**
  - *Mutation:* Schließe bereits vorhandene Allele bei der zufälligen Wahl aus.
  - *Crossover:* Stelle zunächst die Feldnummern je Chromosom zusammen, die im jeweils anderen Chromosom nicht vorkommen, und wende auf die so verkürzten Chromosomen das Ein-Punkt-Crossover an.
- **Reparaturmechanismus:** Finde und ersetze doppelt bzw. mehrfach auftretende Feldnummern, so daß alle Feldnummern verschieden werden.
- **Strafterm:** Verringere die Fitneß um die Anzahl der Doppel-/Mehrfachbelegungen von Feldern, ggf. multipliziert mit einem Gewichtungsfaktor.

Genetische Algorithmen: Verlassen des Suchraums

Weiteres Beispiel: **Problem des Handlungsreisenden**

- Eine Rundreise wird durch eine Permutation der Städte dargestellt.
  (Stadt an k-ter Position wird im k-ten Schritt besucht.)
- Ein-Punkt-Crossover kann den Raum der Permutationen verlassen:

3	5	2	8	1	7	6	4
1	2	3	4	5	6	7	8

3	5	2	4	5	6	7	8
1	2	3	8	1	7	6	4

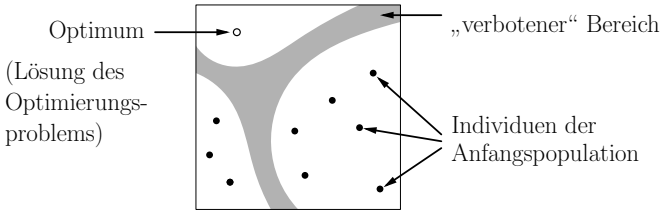
- **Kodierungsspezifische genetische Operatoren:**
  - *Mutation:* z.B. Zweiertausch, Verschieben eines Teilstücks, Inversion
  - *Crossover:* Kantenrekombination (wird später besprochen)
- **Reparaturmechanismus:** Entferne doppelt auftretende Städte und hänge die fehlenden Städte am Ende an:

3	5	2	4	~~5~~	6	7	8	**1**

- **Strafterm:** Verringern der Fitneß um eine Konstante für jede fehlende Stadt.

Genetische Algorithmen: Verlassen des Suchraums

- Bei **nicht zusammenhängenden Suchräumen** können **Reparaturmechanismen** die Suche erschweren, da Lösungskandidaten in den „verbotenen“ Bereichen sofort wieder in den erlaubten Bereich zurückgeführt werden.



- In diesen Fällen ist es angebrachter, einen **Strafterm** einzuführen, durch den Lösungskandidaten im „verbotenen“ Bereich zwar bestraft, aber nicht entfernt werden. Dieser Strafterm sollte im Laufe der Zeit wachsen, um Lösungskandidaten in den „verbotenen“ Bereichen in späteren Generationen zu unterdrücken.

Elemente genetischer Algorithmen

2. Fitneßfunktion und Selektionsverfahren

Genetische Algorithmen: Selektion

- **Prinzip der Selektion:** Bessere Individuen (bessere Fitneß) sollen größere Chancen haben, Nachkommen zu haben (differentielle Reproduktion).

  Die Stärke der Bevorzugung guter Individuen heißt **Selektionsdruck**.

Bei der Wahl des Selektionsdrucks gibt es einen Gegensatz von

- **Durchforstung des Suchraums (exploration)**:
  Die Individuen sollten möglichst breit über den Suchraum gestreut sein, damit die Chancen, daß das globale Optimum gefunden wird, möglichst groß sind.
  → geringer Selektionsdruck wünschenswert
- **Ausbeutung guter Individuen (exploitation):**
  Es sollte das (u.U. lokale) Optimum in der Nähe guter Individuen angestrebt werden (Konvergenz zum Optimum).
  → hoher Selektionsdruck wünschenswert

Genetische Algorithmen: Selektion

Wahl des Selektionsdrucks:

- **Beste Strategie:** Zeitabhängiger Selektionsdruck
  - geringer Selektionsdruck in früheren Generationen
  - höherer Selektionsdruck in späteren Generationen

  → zuerst gute Durchforstung des Suchraums,
  dann Ausbeutung der erfolgversprechendsten Region
- Der Selektionsdruck wird über eine Skalierung der Fitneßfunktion oder über Parameter des Selektionsverfahrens gesteuert.
- Wichtige **Selektionsverfahren** und **Skalierungsmethoden**:
  - Glücksradauswahl
  - Rangauswahl
  - Turnierauswahl
  - Anpassung der Fitneßvariation
  - linear dynamische Skalierung
  - σ-Skalierung

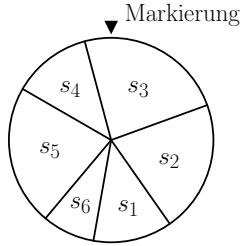
Selektion: Glücksradauswahl

- **Glücksradauswahl** (roulette wheel selection) ist das bekannteste Verfahren.
- Berechne die relative Fitneß der Individuen:

$$f_{\text{rel}}(s) = \frac{f_{\text{abs}}(s)}{\sum_{s' \in \text{pop}(t)} f_{\text{abs}}(s')}$$

  und interpretiere sie als Auswahlwahrscheinlichkeit eines Individuums (sogenannte **fitneßproportionale Selektion**).
- **Beachte:** Die absolute Fitneß $f_{\text{abs}}(s)$ darf in diesem Fall nicht negativ sein; ggf. ist ein positiver Wert zu addieren oder negative Werte sind Null zu setzen.
- **Beachte:** Die Fitneß muß zu maximieren sein. (Sonst würden schlechte Individuen mit hoher Wahrscheinlichkeit gewählt).
- **Veranschaulichung:** Glücksrad mit einem Sektor je Individuum s. Die Sektorgrößen entsprechen den relativen Fitneßwerten $f_{\text{rel}}(s)$.

Selektion: Glücksradauswahl



Auswahl eines Individuums:

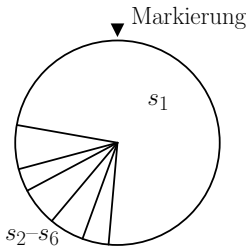
- Drehe Glücksrad.
- Wähle Chromosom, dessen Sektor an der Markierung liegt.

Auswahl der Zwischenpopulation:

- Wiederhole die Auswahl so oft, wie es Individuen in der Population gibt.

- **Technischer Nachteil der Glücksradauswahl**:
  Zur Berechnung der relativen Fitneß müssen die Fitneßwerte aller Individuen summiert werden (Normierungsfaktor).
  - Ausgangspopulation muß während der Auswahl konstant bleiben.
  - Parallelisierung der Implementierung wird erschwert.

Glücksradauswahl: Dominanzproblem



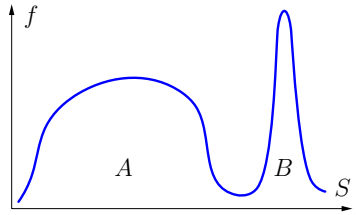
- Hat ein Individuum eine sehr hohe Fitneß, kann es die Auswahl **dominieren**.
- In den Folgegenerationen wird diese Dominanz noch verstärkt, da dann Kopien und sehr ähnliche Individuen vorliegen.
- Als Ergebnis kommt es zum **Crowding**: Die Population besteht aus gleichen und sehr ähnlichen Individuen.

- Crowding führt dazu, daß das (lokale) Optimum sehr schnell gefunden wird.
- **Nachteil:** Die Diversität der Population geht verloren.
  - Ausbeutung eines oder weniger guter Individuen.
  - Keine Durchforstung des Suchraums, sondern lokale Optimierung. (in späten Generationen erwünscht, am Anfang unerwünscht.)

Selektion: Einfluß der Fitneßfunktion

Das Dominanzproblem weißt auf den starken Einfluß der Fitneßfunktion auf die Wirkung der fitneßproportionalen Selektion hin.

- Problem der **vorzeitigen Konvergenz**:
  Nimmt die zu maximierende Funktion stark unterschiedliche Werte an, so kann es zu vorzeitiger Konvergenz kommen.
- Beispiel: Ist am Anfang im Bereich B kein Chromosom, so bleibt die Population durch die Selektion in der Nähe des (lokalen) Maximums im Bereich A.

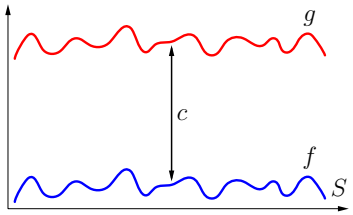


Individuen, die sich dem Übergangsbereich zwischen A und B nähern, haben nur sehr schlechte Chancen auf Nachkommen.

Selektion: Einfluß der Fitneßfunktion

Allgemein stellt sich das Problem der **absoluten Höhe** der Fitneßwerte im Vergleich zu ihrer **Variation** (Spannweite), denn es gibt umgekehrt auch das

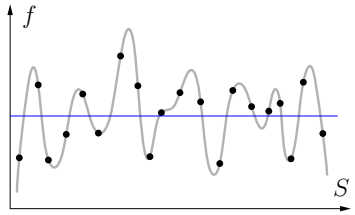
- Problem des **verschwindenden Selektionsdrucks**:
  - Die Maximierung einer Funktion $f : S \to \mathbb{R}$ ist äquivalent zur Maximierung einer Funktion $g : S \to \mathbb{R}$ mit $g(s) \equiv f(s) + c$, $c \in \mathbb{R}$.
  - Falls $c \gg \sup_{s \in S} f(s)$, so folgt $\forall s \in S : g_{\text{rel}}(s) \approx \frac{1}{\text{popsize}}$.
    $\to$ (zu) geringer Selektionsdruck



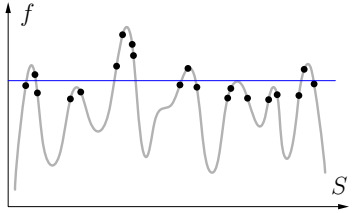
Obwohl die Maxima an den gleichen Stellen liegen, sind sie mit einem genetischen Algorithmus unterschiedlich leicht zu finden: mit g gibt es nur (zu) geringe Unterschiede der relativen Fitneß.

Selektion: Abnehmender Selektionsdruck

- Da ein genetischer Algorithmus die (durchschnittliche) Fitneß der Individuen tendenziell von Generation zu Generation steigert, erzeugt er u.U. selbst das Problem des verschwindenden Selektionsdrucks.
- Höherer Selektionsdruck am Anfang, da die Fitneßwerte zufällig verteilt sind, geringerer Selektionsdruck in späteren Generationen (umgekehrt wäre besser).
- Beispiel: Die Punkte zeigen die Individuen der Generation.



frühe Generation



späte Generation

Selektion: Anpassung der Fitneßfunktion

Lösungsansatz für die angesprochenen Probleme: **Skalierung der Fitneß**

- **Linear dynamische Skalierung**

$$f_{\text{lds}}(s) = \alpha f(s) - \min\{f(s') \mid s' \in \text{pop}(t)\}, \qquad \alpha > 0.$$

  Statt des Minimums der aktuellen Population $\text{pop}(t)$ wird auch das Minimum der letzten k Generationen benutzt. Gewöhnlich ist $\alpha > 1$.

- **σ-Skalierung**

$$f_\sigma(s) = f(s) - (\mu_f(t) - \beta \cdot \sigma_f(t)), \qquad \beta > 0.$$

$$\mu_f(t) = \frac{1}{\text{popsize}} \sum_{s \in \text{pop}(t)} f(s) \qquad \text{(Mittelwert der Fitneß)}$$

$$\sigma_f(t) = \sqrt{\frac{1}{\text{popsize} - 1} \sum_{s \in \text{pop}(t)} (f(s) - \mu_f(t))^2} \qquad \text{(Standardabweichung)}$$

- Problem: Wahl der Parameter α und β.

Selektion: Anpassung der Fitneßfunktion

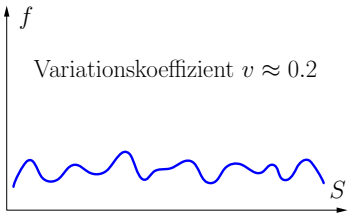
- Betrachte den **Variationskoeffizienten** der Fitneßfunktion

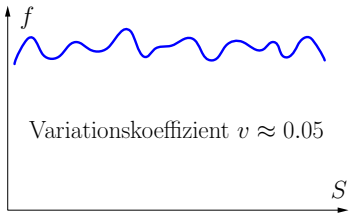
$$v = \frac{\sigma_f}{\mu_f} = \frac{\sqrt{\frac{1}{|S|-1} \sum_{s' \in S} \left(f(s') - \frac{1}{|S|} \sum_{s \in S} f(s) \right)^2}}{\frac{1}{|S|} \sum_{s \in S} f(s)} \qquad \text{bzw.} \quad v(t) = \frac{\sigma_f(t)}{\mu_f(t)}.$$

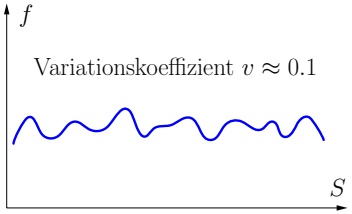
- Empirisch wurde festgestellt, daß ein Variationskoeffizient $v \approx 0.1$ ein gutes Verhältnis von Durchforstung und Ausbeutung liefert.
- Weicht v von diesem Wert ab, so versucht man, diesen Wert durch Anpassung der Fitneßfunktion f zu erreichen (z.B. durch Skalierung, Exponentiation).
- Für die praktische Berechnung von v wird der Lösungsraum S durch die aktuelle Population $\text{pop}(t)$ ersetzt (v ist nicht berechen-, sondern nur schätzbar).
- Es werden dann in jeder Generation der Wert von $v(t)$ berechnet und die Fitneßwerte entsprechend angepaßt (σ-Skalierung mit $\beta = \frac{1}{v^*}$, $v^* = 0.1$).

Selektion: Anpassung der Fitneßfunktion

Illustration des Variationskoeffizienten:







Bei zu hohem Variationskoeffizienten kommt es zu vorzeitiger Konvergenz, bei zu niedrigem zu verschwindendem Selektionsdruck. $v \approx 0.1$ ist erfahrungsgemäß ein guter Wert.

Selektion: Anpassung der Fitneßfunktion

- **Zeitabhängige Fitneßfunktion:**
  Bestimme die relative Fitneß nicht direkt aus der zu optimierenden Funktion $f(s)$, sondern aus $g(s) \equiv (f(s))^{k(t)}$.

  Der zeitabhängige Exponent $k(t)$ steuert den Selektionsdruck.

- **Verfahren zur Bestimmung von** $k(t)$ [Michalewicz 1996]:
  (soll den Variationskoeffizienten v in der Nähe von $v^* \approx 0.1$ halten)

$$k(t) = \left(\frac{v^*}{v}\right)^{\beta_1} \left(\tan\left(\frac{t}{T+1} \cdot \frac{\pi}{2}\right)\right)^{\beta_2 \left(\frac{v}{v^*}\right)^{\alpha}}$$

v^*, β_1, β_2, α	Parameter des Verfahrens
v	Variationskoeffizient (z.B. aus Anfangspopulation geschätzt)
T	maximale Anzahl zu berechnender Generationen
t	aktueller Zeitschritt (Nummer der Generation)

Empfehlung: $v^* = 0.1$, $\beta_1 = 0.05$, $\beta_2 = 0.1$, $\alpha = 0.1$

Selektion: Anpassung der Fitneßfunktion

- **Boltzmann-Selektion** (andere zeitabhängige Fitneßfunktion):
  Bestimme die relative Fitneß nicht direkt aus der zu optimierenden Funktion $f(s)$, sondern aus $g(s) \equiv \exp\left(\frac{f(s)}{kT}\right)$.

  Der zeitabhängige **Temperaturparameter** T steuert den Selektionsdruck. k ist eine Normierungskonstante.

  Die Temperatur kann z.B. linear bis zu einer vorher festgelegten Maximalzahl an Generationen abnehmen.

- Die Idee dieses Auswahlverfahrens ähnelt der des **simulierten Ausglühens**:
  - In frühen Generationen ist der Temperaturparameter hoch, die relativen Unterschiede zwischen den Fitneßwerten daher gering.
  - In späteren Generationen wird der Temperaturparameter gesenkt, wodurch die Fitneßunterschiede größer werden.
  - **Folge:** Im Laufe der Generationen steigt der Selektionsdruck.

Glücksradauswahl: Varianzproblem

- Die Auswahl der Individuen ist zwar fitneßproportional, aber dennoch zufällig.
- Es gibt keine Garantie, daß gute Individuen in die nächste Generation kommen, nicht einmal für das beste Individuum.
- Allgemein: Die Zahl der Nachkommen eines Individuums kann stark vom Erwartungswert abweichen (**hohe Varianz der Nachkommenzahl**).

  (Berechnung des Erwartungswertes: siehe Übungsaufgabe)

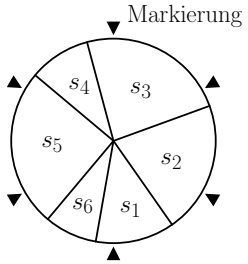
- Sehr einfache, aber nicht unbedingt empfehlenswerte Lösung:
  Diskretisierung des Fitneßwertebereichs
  - Berechne Mittelwert $\mu_f(t)$ und Standardabweichung $\sigma_f(t)$ der Population.
  - Wenn $\mu_f(t) - \sigma_f(t) > f(s)$: 0 Nachkommen
  - Wenn $\mu_f(t) - \sigma_f(t) \leq f(s) \leq \mu_f(t) + \sigma_f(t)$: 1 Nachkomme
  - Wenn $f(s) > \mu_f(t) + \sigma_f(t)$: 2 Nachkommen

Selektion: Erwartungswertmodell

Lösungsmöglichkeit für das Varianzproblem: **Erwartungswertmodell**

- Erzeuge für jeden Lösungskandidaten $\lfloor f_{\text{rel}}(s) \cdot \text{popsize} \rfloor$ Individuen.
- Fülle die (Zwischen-)Population durch Glücksradauswahl auf.

Alternative: **Stochastic Universal Sampling**



Auswahl der Zwischenpopulation:

- Drehe Glücksrad einmal.
- Wähle ein Chromosom je Markierung.
- Hier: $1 \times s_1$, $1 \times s_2$, $2 \times s_3$, $2 \times s_5$.
- Überdurchschnittlich gute Individuen kommen sicher in die Zwischenpopulation.

Selektion: Erwartungswertmodell

Varianten des Erwartungswertmodells:

Erzeugen der restlichen Individuen der (Zwischen-)Population durch

- Verfahren, die aus der **Wahlauswertung** bekannt sind (Mandats-/Sitzverteilung, z.B. größte Reste, Hare-Niemeyer, d'Hondt etc.)
- **Glücksradauswahl**, aber:
  - Für jedes Individuum, daß einen Nachkommen erhält, wird seine Fitneß um einen bestimmten Betrag Δf verringert.
  - Wird die Fitneß eines Individuums dadurch negativ, so erhält es keine weiteren Nachkommen.
  - Prinzip für die Wahl Δf: Das beste Individuum erhält höchstens eine festgelegte Zahl k von Nachkommen:

$$\Delta f = \frac{1}{k} \max\{f(s) \mid s \in \text{pop}(t)\}.$$

Selektion: Rangauswahl

- Die Individuen werden nach ihrer Fitneß absteigend sortiert. So erhält jedes Individuum einen **Rang** in der Population. (Idee aus der Statistik: verteilungsfreie Verfahren, z.B. Rangkorrelation)
- Über der Rangskala wird eine Wahrscheinlichkeitsverteilung definiert: Je kleiner die Rangnummer, desto größer die Wahrscheinlichkeit.
- Anschließend wird mit dieser Verteilung eine Glücksradauswahl durchgeführt.
- **Vorteile:**
  - Das Dominanzproblem kann weitgehend vermieden werden, da der Wert der Fitneßfunktion nicht direkt die Auswahlwahrscheinlichkeit beeinflußt.
  - Über die Wahrscheinlichkeitsverteilung auf der Rangskala kann der Selektionsdruck sehr bequem gesteuert werden.
- **Nachteil:**
  - Die Individuen müssen sortiert werden (Aufwand: $\text{popsize} \cdot \log \text{popsize}$).

Selektion: Turnierauswahl

- Für die Auswahl eines Individuums werden k, $2 \leq k < \text{popsize}$, Individuen zufällig aus der Population gezogen (mit oder ohne Zurücklegen, Auswahl *ohne* Berücksichtigung der Fitneß, k ist die **Turniergröße**).
- Die Individuen tragen ein Turnier aus, das das beste Individuum gewinnt. Der Turniersieger erhält einen Nachkommen in der Zwischenpopulation.
- Anschließend werden *alle* Individuen des Turniers (auch der Sieger) in die aktuelle Population zurückgelegt.
- **Vorteile:**
  - Das Dominanzproblem kann weitgehend vermieden werden, da der Wert der Fitneßfunktion nicht direkt die Auswahlwahrscheinlichkeit beeinflußt.
  - Über die Turniergröße kann m.E. der Selektionsdruck gesteuert werden.
- **Modifikation:** Die relative Fitneß der Turnierteilnehmer bestimmt die Gewinnwahrscheinlichkeit (Glücksradauswahl eines Individuums im Turnier).

Selektion: Elitismus

- Nur im Erwartungswertmodell (oder einer seiner Varianten) ist sichergestellt, daß das beste Individuum in die Zwischenpopulation gelangt.
- Auch wenn das beste Individuum in die Zwischenpopulation gelangt, ist es nicht vor Veränderungen durch genetische Operatoren geschützt.
  (Dies gilt auch für das Erwartungswertmodell.)
- **Folglich:** Die Fitneß des besten Individuums kann von einer Generation zur nächsten auch wieder abnehmen. (Das ist natürlich unerwünscht.)
- **Lösung: Elitismus**
  Das beste Individuum (oder die k, $1 \leq k <$ popsize, besten Individuen) werden *unverändert* in die nächste Generation übernommen.
  (Die *Elite* einer Population bleibt erhalten, daher *Elitismus*.)
- **Beachte:** Die Elite wird *nicht* von der normalen Auswahl ausgenommen, da sie ja auch noch durch genetische Operatoren verbessert werden kann.

Selektion: Elitismus

- Meist ersetzen Nachkommen (Mutations-/Crossover-Produkte) ihre Eltern.
- **„lokaler“ Elitismus** (Elitismus zwischen Eltern und Nachkommen)
  - *Mutation:* Ein mutiertes Individuum ersetzt seinen Elter nur dann, wenn es eine mindestens ebenso gute Fitneß besitzt.
  - *Crossover:* Die vier am Crossover beteiligten Individuen (zwei Eltern und zwei Nachkommen) werden nach Fitneß sortiert. Die beiden besten Individuen werden in die nächste Generation übernommen.
- **Vorteil**
  - Bessere Konvergenzeigenschaften,
    da das lokale Optimum konsequenter angestrebt wird.
- **Nachteil**
  - Relativ große Gefahr des Hängenbleibens in lokalen Optima,
    da keine (lokalen) Verschlechterungen möglich sind.

Selektion: Nischentechniken

Ziel von Nischentechniken: **Explizites Vermeiden des Crowding**

- **Deterministisches Crowding**
  - *Idee:* Erzeugte Nachkommen ersetzen stets die ihnen ähnlichsten Individuen der Population → Suchraum wird lokal weniger dicht besetzt.
  - *Benötigt:* ein Ähnlichkeits- oder Abstandsmaß für die Individuen
    (bei binärkodierten Chromosomen z.B. der Hammingabstand)
- **Variante des deterministischen Crowding**
  - Beim Crossover werden zwei Paare von Individuen aus je einem Elter und einem Nachkommen gebildet, wobei ein Nachkomme dem ihm ähnlichsten Elter zugeordnet wird.
  - Aus jedem Paar wird das bessere Individuum übernommen.
  - *Vorteil:* Es sind deutlich weniger Vergleiche zwischen Individuen nötig,
    da nicht die ganze Population betrachtet wird.

Selektion: Nischentechniken

- **Sharing**
  - *Idee:* Die Fitneß eines Individuums wird reduziert, wenn sich in seiner Nachbarschaft noch weitere Individuen befinden.
    Anschaulich: **Die Individuen teilen sich die Fitneß einer Nische.**
  - *Benötigt:* ein Ähnlichkeits- oder Abstandsmaß für die Individuen
  - *Beispiel:*

$$f_{\text{share}}(s) = \frac{f(s)}{\sum_{s' \in \text{pop}(t)} g(d(s, s'))}$$

d: Abstandsmaß für die Individuen
g: Wichtungsfunktion, die die Form und Größe der Nische definiert,
z.B. sogenanntes **power law sharing**:

$$g(x) = \begin{cases} 1 - \left(\frac{x}{\varrho}\right)^{\alpha}, & \text{falls } x < \varrho, \\ 0, & \text{sonst.} \end{cases}$$

ϱ: Nischenradius, α: steuert die Einflußstärke innerhalb der Nische

Selektionsverfahren: Charakterisierung

Selektionsverfahren werden gern mit folgenden Begriffspaaren charakterisiert:

statisch	Die Auswahlwahrscheinlichkeiten bleiben konstant.
dynamisch	Die Auswahlwahrscheinlichkeiten ändern sich.
extinctive	(auslöschend) Auswahlwahrscheinlichkeiten dürfen 0 sein.
preservative	(erhaltend) Alle Auswahlwahrscheinlichkeiten müssen positiv sein.
rein	Individuen dürfen nur in einer Generation Nachkommen haben.
unrein	Individuen dürfen in mehreren Generationen Nachkommen haben.
rechts	Alle Individuen einer Population dürfen sich vermehren.
links	Die besten Individuen einer Population dürfen sich *nicht* vermehren (um vorzeitige Konvergenz zu vermeiden).
generational	Die Eltermenge ist fest, bis alle Nachkommen erzeugt sind.
on the fly	Erzeugte Nachkommen ersetzen unmittelbar ihre Eltern.

Elemente genetischer Algorithmen

3. Genetische Operatoren

Genetische Operatoren

- Auf einen Teil der ausgewählten Individuen (Zwischenpopulation) werden **genetische Operatoren** angewandt, um Varianten und Rekombinationen der bestehenden Lösungskandidaten zu erzeugen.
- Allgemeine Einteilung genetischer Operatoren nach der Zahl der Eltern:
  - Ein-Elter-Operatoren („Mutation“)
  - Zwei-Elter-Operatoren („Crossover“)
  - Mehr-Elter-Operatoren
- Je nach verwendeter Kodierung müssen die genetischen Operatoren bestimmte Eigenschaften haben, z.B.:
  - Sind die Lösungskandidaten durch Permutationen kodiert, so sollten die genetischen Operatoren permutationserhaltend sein.
  - Allgemein: Sind bestimmte Allelkombinationen unsinnig, sollten die genetischen Operatoren sie möglichst nicht erzeugen.

Genetische Ein-Elter-Operatoren I

Genetische Ein-Elter-Operatoren bezeichnen wir auch allgemein als **Mutation**.

- **Standardmutation**
  Austausch der Ausprägung eines Gens durch eine andere.

3	1	4	2	5	4	6	→	3	1	6	2	5	4	6

  - Gegebenfalls werden mehrere Gene mutiert (vgl. n-Damen-Problem).
  - *Parameter:* Mutationswahrscheinlichkeit p_m, $0 < p_m \ll 1$
    Für Bitstrings ist $p_m = \frac{1}{\text{length}(s)}$ annähernd optimal.
- **Zweiertausch**
  Austausch der Ausprägungen zweier Gene eines Chromosoms.

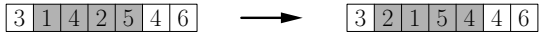
3	1	4	2	5	4	6	→	3	5	4	2	1	4	6

  - *Voraussetzung:* gleiche Allelmengen der ausgetauschten Gene.
  - *Verallgemeinerung:* zyklischer Tausch von 3, 4, ..., k Genen.

Genetische Ein-Elter-Operatoren II

- **Verschieben eines Teilstücks**

- **Mischen/Permutation eines Teilstücks**



- **Inversion (Umdrehen eines Teilstücks)**

- *Voraussetzung:* gleiche Allelmengen im betroffenen Bereich.
- *Parameter:* Ggf. Wahrscheinlichkeitsverteilung über Längen (und Verschiebungsweiten für Verschieben eines Teilstücks).

Genetische Zwei-Elter-Operatoren I

Genetische Zwei-Elter-Operatoren bezeichnen wir auch allgemein als **Crossover**.

- **Ein-Punkt-Crossover**
  - Bestimmen eines zufälligen Schnittpunktes
  - Austausch der Gensequenzen auf einer Seite des Schnittpunktes

- **Zwei-Punkt-Crossover**
  - Bestimmen zweier zufälliger Schnittpunkte
  - Austausch der Gensequenzen zwischen den beiden Schnittpunkten

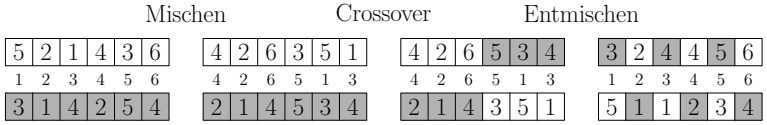
Genetische Zwei-Elter-Operatoren II

- **n-Punkt-Crossover**
  - Verallgemeinerung des Ein- und Zwei-Punkt-Crossover
  - Bestimmen von n zufälligen Schnittpunkten
  - Abwechselndes Austauschen / Nicht-Austauschen der Gensequenzen zwischen zwei aufeinanderfolgenden Schnittpunkten
- **Uniformes Crossover**
  - Für jedes Gen wird einzeln bestimmt, ob es ausgetauscht wird oder nicht (+: ja, −: nein, *Parameter:* Wahrscheinlichkeit p_x für Austausch).

  - **Beachte:** Das uniforme Crossover entspricht **nicht** dem (length(s) − 1)-Punkt-Crossover! Die Zahl der Crossoverpunkte wird zufällig bestimmt.

Genetische Zwei-Elter-Operatoren III

- **Shuffle Crossover**
  - Bevor Ein-Punkt-Crossover angewandt wird, werden die Gene zufällig gemischt, nach dem Crossover wieder entmischt.

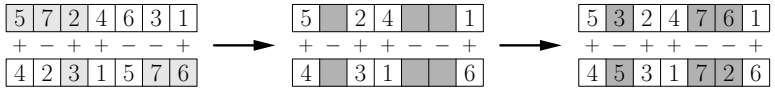


  - **Beachte:**
    Das Shuffle Crossover ist **nicht** äquivalent zum uniformen Crossover!
  - Beim Shuffle Crossover ist jede Anzahl von Vertauschungen von Genen zwischen den Chromosomen gleichwahrscheinlich, beim uniformen Crossover ist die Anzahl binomialverteilt mit dem Parameter p_x.
  - Das Shuffle Crossover ist eines der empfehlenswertesten Verfahren.

Genetische Zwei-Elter-Operatoren IV

Permutationserhaltende Crossover-Operatoren

- **Uniformes ordnungsbasiertes Crossover**
  - Ähnlich wie beim normalen uniformen Crossover wird für jedes Gen einzeln entschieden, ob es erhalten bleibt oder nicht
    (+: ja, −: nein, *Parameter:* Wahrscheinlichkeit p_k für Erhalt).
  - Die Lücken werden durch die fehlenden Allele aufgefüllt, und zwar in der Reihenfolge, in der sie im *anderen* Chromosom vorkommen.



  - Dieses Verfahren erhält **Reihenfolgeinformation**.
  - *Alternative:* In einem Chromosom werden die durch „+“ markierten, im anderen die durch „−“ markierten Gene erhalten.

Genetische Zwei-Elter-Operatoren V

Permutationserhaltende Crossover-Operatoren

- **Kantenrekombination** (speziell für Problem des Handlungsreisenden)
  - Chromosom wird als Graph (Kette oder Ring) aufgefaßt:
    Jedes Gen besitzt Kanten zu seinen Nachbarn im Chromosom.
  - Die Kanten der Graphen zweier Chromosomen werden gemischt, daher der Name *Kantenrekombination*.
  - Dieses Verfahren erhält **Nachbarschaftsinformation**.
- **Vorgehen:**
  1. *Aufbau einer Kantentabelle:*
     - Zu jedem Allel werden seine Nachbarn (in beiden Eltern) aufgelistet. (Ggf. sind das erste und das letzte Gen des Chromosoms benachbart.)
     - Hat ein Allel in beiden Eltern den gleichen Nachbarn (Seite irrelevant), so wird dieser Nachbar nur einmal aufgeführt, dafür aber markiert.

Zwei-Elter-Operatoren: Kantenrekombination

- **Vorgehen:**
  2. *Aufbau eines Nachkommen:*
     - Das erste Allel wird zufällig aus einem der beiden Eltern gewählt.
     - Ein ausgewähltes Allel wird aus der Kantentabelle gelöscht (aus den Listen der Nachbarn der Allele).
     - Das jeweils nächste Allel wird aus den noch nicht gelöschten Nachbarn des vorangehenden gewählt, wobei die folgende **Prioritätsliste** gilt:
       a) markierte (d.h. doppelt auftretende) Nachbarn
       b) Nachbarn mit kürzester Nachbarschaftsliste (wobei markierte Nachbarn einfach zählen)
       c) zufällige Auswahl eines Nachbarn
- Ein zweiter Nachkomme kann analog aus dem ersten Allel des anderen Elter erzeugt werden. (Dies wird jedoch meist nicht gemacht.)

Zwei-Elter-Operatoren: Kantenrekombination

Beispiel: **A:** 6 3 1 5 2 7 4 **B:** 3 7 2 5 6 1 4

Aufbau der Kantentabelle

Allel	Nachbarn in **A**	Nachbarn in **B**	zusammengefaßt
1	3, 5	6, 4	3, 4, 5, 6
2	5, 7	7, 5	5*, 7*
3	6, 1	4, 7	1, 4, 6, 7
4	7, 6	1, 3	1, 3, 6, 7
5	1, 2	2, 6	1, 2*, 6
6	4, 3	5, 1	1, 3, 4, 5
7	2, 4	3, 2	2*, 3, 4

- Beide Chromosomen werden als Ring aufgefaßt (erstes und letztes Gen sind benachbart):

  In **A** ist der linke Nachbar der 6 die 4, der rechte Nachbar der 4 die 6; in **B** analog.
- In beiden Chromosomen stehen 5, 2 und 7 nebeneinander — das sollte erhalten werden (zeigt sich in Markierungen).

Zwei-Elter-Operatoren: Kantenrekombination

Aufbau eines Nachkommen

6	5	2	7	4	3	1

Allel	Nachbarn	Wahl: 6	5	2	7	4	3	1
1	3, 4, 5, 6	3, 4, 5	3, 4	3, 4	3, 4	3		
2	5*, 7*	5*, 7*	7*	**7***	—	—	—	—
3	1, 4, 6, 7	1, 4, 7	1, 4, 7	1, 4, 7	1, 4	1	**1**	—
4	1, 3, 6, 7	1, 3, 7	1, 3, 7	1, 3, 7	1, 3	**1, 3**	—	—
5	1, 2*, 6	1, 2*	**1, 2***	—	—	—	—	—
6	1, 3, 4, 5	**1, 3, 4, 5**	—	—	—	—	—	—
7	2*, 3, 4	2*, 3, 4	2*, 3, 4	3, 4	**3, 4**	—	—	—

- Starte mit erstem Allel des Chromosoms **A**, also der 6, und streiche die 6 aus allen Nachbarschaftslisten (dritte Spalte).
- Da unter den Nachbarn der 6 (das sind 1, 3, 4, 5) die 5 die kürzeste Liste hat, wird die 5 für das zweites Gen gewählt. Dann folgt die 2, die 7 usw.

Zwei-Elter-Operatoren: Kantenrekombination

- Der Nachkomme hat meist eine neue Kante (vom letzten zum ersten Gen).
- Die Kantenrekombination kann auch angewendet werden, wenn das erste und das letzte Gen eines Chromosoms nicht als benachbart angesehen werden: Die entsprechenden Kanten werden dann nicht in die Kantentabelle aufgenommen.
- Werden erstes und letztes Gen als benachbart angesehen, kann das Startallel beliebig aus den Chromosomen gewählt werden. Werden sie dagegen nicht als benachbart angesehen, muß es ein am Anfang stehendes Allel sein.
- Beim Aufbau eines Nachkommen kann es vorkommen, daß die Nachbarschaftsliste des gerade ausgewählten Allels leer ist.

  (Die Prioritätsregeln dienen dazu, die Wahrscheinlichkeit dafür gering zu halten; sie sind aber nicht perfekt.)

  In diesem Fall wird die Konstruktion mit einem Allel fortgesetzt, das zufällig aus den noch übrigen Allelen gewählt wird.

Genetische Drei- und Mehr-Elter-Operatoren

- **Diagonal-Crossover**
  - Ähnlich wie Ein-, Zwei- und n-Punkt-Crossover, aber für mehr Eltern.
  - Bei drei Eltern werden zwei Crossover-Punkte gewählt.
  - An den Schnittstellen werden die Gensequenzen diagonal und zyklisch über die Chromosomen verschoben.

  - Eine Verallgemeinerung auf mehr als drei Eltern ist naheliegend. Für k Eltern werden $k-1$ Crossover-Punkte gewählt.
  - Dieses Verfahren führt zu einer sehr guten Durchforstung des Suchraums, besonders bei großer Elternzahl (10–15 Eltern).

Charakterisierung von Crossover-Operatoren

- **Ortsabhängige Verzerrung (positional bias)**
  - Liegt vor, wenn die Wahrscheinlichkeit, daß zwei Gene zusammen vererbt werden (im gleichen Chromosom bleiben, zusammen in das andere Chromosom wandern), von ihrer relativen Lage im Chromosom abhängt.
  - Ist unerwünscht, da dann die Anordnung der Gene im Chromosom entscheidenden Einfluß auf den Erfolg oder Mißerfolg des genetischen Algorithmus haben kann (bestimmte Anordnungen lassen sich schwerer erreichen).
- **Beispiel: Ein-Punkt-Crossover**
  - Zwei Gene werden voneinander getrennt (gelangen in verschiedene Nachkommen), wenn der Crossover-Punkt zwischen sie fällt.
  - Je näher zwei Gene im Chromosom beieinander liegen, desto so weniger mögliche Crossover-Punkte gibt es zwischen ihnen.
  - *Folglich:* Nebeneinander liegende Gene werden mit höherer Wahrscheinlichkeit als entfernt liegende in den gleichen Nachkommen gelangen.

Charakterisierung von Crossover-Operatoren

- **Verteilungsverzerrung (distributional bias)**
  - Liegt vor, wenn die Wahrscheinlichkeit, daß eine bestimmte Anzahl von Genen ausgetauscht wird, nicht für alle Anzahlen gleich ist.
  - Ist oft unerwünscht, da dann Teillösungen unterschiedlicher Größe unterschiedlich gute Chancen haben, in die nächste Generation zu gelangen.
  - Die Verteilungsverzerrung ist meist weniger kritisch (d.h. eher tolerierbar) als die ortabhängige Verzerrung.
- **Beispiel: uniformes Crossover**
  - Da jedes Gen unabhängig von allen anderen mit der Wahrscheinlichkeit p_x ausgetauscht wird, ist die Anzahl k der ausgetauschten Gene binomialverteilt mit dem Parameter p_x:

    $P(K = k) = \binom{n}{k} p_x^k (1 - p_x)^{n-k}$ mit $n \mathrel{\hat{=}}$ Gesamtzahl der Gene.
  - *Folglich:* Sehr kleine und sehr große Anzahlen sind unwahrscheinlicher.

Theoretische Betrachtung:

Das Schematheorem

Das Schematheorem

- Frage: **Warum funktionieren genetische Algorithmen?**
- **Ansatz** von [Holland 1975]:

  Betrachte Chromosomenschemata (d.s. nur teilweise festgelegte Chromosomen) und untersuche, wie sich die Zahl der Chromosomen, die zu einem Schema passen, über die Generationen hinweg entwickelt.
- **Ziel:** Eine zumindest grobe stochastische Aussage darüber, wie genetische Algorithmen den Suchraum durchforsten.
- Zur **Vereinfachung** der Darstellung: Beschränkung auf
  - Bitfolgen (Chromosomen aus Nullen und Einsen) mit fester Länge L
  - Fitneßproportionale Selektion (Glücksradauswahl)
  - Standardmutation (Änderung eines zufällig gewählten Bits)
  - Ein-Punkt-Crossover (Durchschneiden an einer Stelle und Vertauschen)

Das Schematheorem: Schemata

- **Definition: Schema**

  Ein **Schema** h ist eine Zeichenkette der Länge L über dem Alphabet $\{0, 1, *\}$, d.h. $h \in \{0, 1, *\}^L$.
  Das Zeichen $*$ heißt **Jokerzeichen** oder **Don't-Care-Symbol**.
- **Definition: Passung**

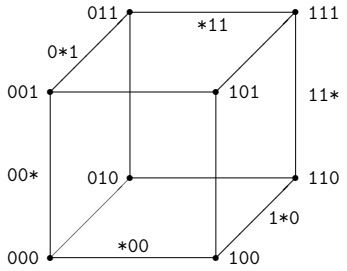
  Ein Chromosom $c \in \{0, 1\}^L$ **paßt zu einem Schema** $h \in \{0, 1, *\}^L$, in Zeichen: $c \triangleleft h$, wenn es mit h an allen Stellen übereinstimmt, an denen h eine 0 oder eine 1 enthält. (Stellen, an denen ein $*$ steht, bleiben unberücksichtigt.)
- **Beispiel:**

$h =$	`**0*11*10*`	Schema der Länge 10	
$c_1 =$	`1100111100`	paßt zu h,	also $c_1 \triangleleft h$
$c_2 =$	`1111111111`	paßt nicht zu h,	also $c_2 \not\triangleleft h$
- Es gibt 2^L Chromosomen und 3^L Schemata.

  Jedes Chromosom paßt zu $\sum_{i=0}^{L} \binom{L}{i} = 2^L$ Schemata.

Schemata: Hyperebenen

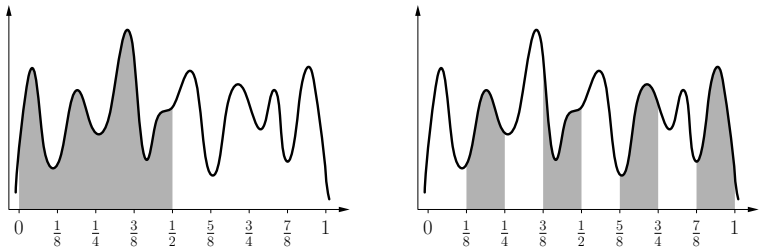
- Jedes Schema beschreibt eine Hyperebene in einem Hypereinheitswürfel (allerdings nur Ebenen, die parallel oder senkrecht zu den Achsen stehen).



- **Beispiele:** `*00` $\hat{=}$ Kante von `000` nach `100` (vorne unten)
  `0**` $\hat{=}$ linke Würfelfläche
  `***` $\hat{=}$ gesamter Würfel

Schemata: Wertebereiche von Funktionen

- Gegeben: reelle Funktion $f : [0, 1] \to \mathbb{R}$
- Annahme: Binärkodierung der Funktionsargumente (kein Gray-Kode)
- Jedem Schema entspricht ein „Streifenmuster“ im Definitionsbereich von f:



Schema `0**...*` Schema `**1*...*`

- Schemata mit Gray-Kodierung: siehe Übungsaufgabe

Das Schematheorem: Einfluß der Selektion

- Um die Verbreitung von Chromosomen, die zu einem Schema passen, verfolgen zu können, müssen wir untersuchen, wie sich die **Selektion** und die **genetischen Operatoren** (Mutation und Crossover) auswirken.
- Für die Selektion müssen wir untersuchen, welche Fitneß Chromosomen haben, die zum Schema h passen. Ansatz: Mittelung über alle Chromosomen.
- **Definition: Mittlere Fitneß**
  Die **mittlere relative Fitneß** der Chromosomen, die in der Generation $\text{pop}(t)$ zum Schema h passen, ist
  $$f_{\text{rel}}(h) = \frac{\sum_{c \in \text{pop}(t), c \lhd h} f_{\text{rel}}(c)}{|\{c \in \text{pop}(t) \mid c \lhd h\}|}.$$
- Die durchschnittliche Anzahl Nachkommen eines zu einem Schema h passenden Chromosoms ist $f_{\text{rel}}(h) \cdot \text{popsize}$.
  Die zu erwartende Zahl Chromosomen, die nach der Auswahl zu einem Schema h passen, ist daher: (Zahl vorher passender Chromosomen) $\cdot f_{\text{rel}}(h) \cdot \text{popsize}$.

Das Schematheorem: Einfluß der Selektion

- Weitere Betrachtungen zur relativen Fitneß eines Schemas:
  $$\begin{aligned} f_{\text{rel}}(h) \cdot \text{popsize} &= \frac{\sum_{c \in \text{pop}(t), c \lhd h} f_{\text{rel}}(c)}{|\{c \in \text{pop}(t) \mid c \lhd h\}|} \cdot \text{popsize} \\ &= \frac{\sum_{c \in \text{pop}(t), c \lhd h} \frac{f(c)}{\sum_{c' \in \text{pop}(t)} f(c')}}{|\{c \in \text{pop}(t) \mid c \lhd h\}|} \cdot \text{popsize} \\ &= \frac{\frac{\sum_{c \in \text{pop}(t), c \lhd h} f(c)}{|\{c \in \text{pop}(t) \mid c \lhd h\}|}}{\frac{\sum_{c' \in \text{pop}(t)} f(c')}{\text{popsize}}} = \frac{\overline{f_t(h)}}{\overline{f_t}} \end{aligned}$$
  $\overline{f_t(h)}$ mittlere Fitneß der in der t-ten Generation zum Schema h passenden Chromsomen
  $\overline{f_t}$ mittlere Fitneß aller Chromosomen der t-ten Generation
- Die mittlere Anzahl Nachkommen kann so durch das Verhältnis der mittleren Güte eines Schemas zur Gesamtdurchschnittsgüte ausgedrückt werden.

Das Schematheorem: Einfluß der Mutation

- Für die genetischen Operatoren brauchen wir Maße, mit denen wir für ein Schema die Wahrscheinlichkeit angeben können, daß durch Anwendung eines Operators die Passung zu diesem Schema verlorengeht bzw. erhalten bleibt.
- **Definition: Ordnung** (für die Standardmutation)

  Die **Ordnung** eines Schemas h ist die Anzahl der Nullen und Einsen in h, also $\text{ord}(h) = \#0 + \#1 = L - \#*$ (#: Anzahl des Auftretens).

  Beispiel: $\text{ord}(\texttt{**0*11*10*}) = 5$.
- Durch eine Standardmutation geht die Passung zu einem Schema verloren mit

  Wahrscheinlichkeit $p_m^*(h) = \frac{\text{ord}(h)}{L}$, falls das Bit umgekehrt wird,

  Wahrscheinlichkeit $p_m^*(h) = \frac{\text{ord}(h)}{2L}$, falls das neue Bit zufällig bestimmt wird.
- **Erläuterung:** Jedes der L Gene eines Chromosoms der Länge L wird mit gleicher Wahscheinlichkeit für eine Mutation ausgewählt.

Das Schematheorem: Einfluß des Crossover

- **Definition: Definierende Länge** (für das Ein-Punkt-Crossover)

  Die **definierende Länge** eines Schemas h ist die Differenz zwischen der Positionsnummer der letzten 0/1 und der Positionsnummer der ersten 0/1 in h.

  Beispiel: $\text{dl}(\texttt{**0*11*10*}) = 9 - 3 = 6$.
- Beim Ein-Punkt-Crossover liegt der Schnittpunkt mit Wahrscheinlichkeit $p_c^*(h) = \frac{\text{dl}(h)}{L-1}$ so, daß zwei Nicht-Jokerzeichen voneinander getrennt werden.
- **Erläuterung:** Bei Chromosomen der Länge L gibt es $L-1$ mögliche Schnittpunkte für das Ein-Punkt-Crossover, die alle gleichwahrscheinlich sind.

  $\text{dl}(h)$ dieser Schnittpunkte liegen so, daß im Schema festgelegte Gene in verschiedene Nachkommen gelangen, wodurch die Passung verlorengehen kann.
- Achtung: Die Passung *kann*, muß jedoch nicht zwangsläufig verlorengehen.

  $\rightarrow$ Für die Rechnung sind weitere Überlegungen nötig (später).

Das Schematheorem: Definitionen

- **Erwartungswert passender Chromosomen**

  $N(h,t)$ ist der Erwartungswert der Anzahl Chromosomen, die in der t-ten Generation zum Schema h passen.
- **Erwartungswert nach Selektion**

  $N(h, t + \Delta t_s)$ ist der Erwartungswert der Anzahl Chromosomen, die in der t-ten Generation nach Selektion zum Schema h passen.
- **Erwartungswert nach Crossover**

  $N(h, t + \Delta t_s + \Delta t_c)$ ist der Erwartungswert der Anzahl Chromosomen, die in der t-ten Generation nach Selektion und Crossover zum Schema h passen.
- **Erwartungswert nach Mutation**

  $N(h, t + \Delta t_s + \Delta t_c + \Delta t_m) = N(h, t+1)$ ist der Erwartungswert der Anzahl Chromosomen, die in der t-ten Generation nach Selektion, Crossover und Mutation (und damit in der $t+1$-ten Generation) zum Schema h passen.

Das Schematheorem: Selektion

- **Gesucht:** Zusammenhang zwischen $N(h,t)$ und $N(h,t+1)$.
- **Vorgehen:** Wir betrachten schrittweise die Auswirkungen der Selektion, des Crossover und der Mutation mit Hilfe der mittleren Fitneß, der Ordnung und der definierenden Länge eines Schemas.
- **Selektion:**

  Die Auswirkungen der Selektion werden durch die mittlere Fitneß beschrieben:

  $$N(h, t + \Delta t_s) = N(h,t) \cdot f_{\text{rel}}(h) \cdot \text{popsize}$$

$N(h,t) \cdot f_{\text{rel}}(h)$	Wahrscheinlichkeit, daß ein zum Schema h passendes Chromosom ausgewählt wird
$f_{\text{rel}}(h) \cdot \text{popsize}$	Durchschnittliche Anzahl Nachkommen eines zum Schema h passenden Chrosomoms
- Beachte: Die relative Fitneß $f_{\text{rel}}(h)$ ist nicht exakt bestimmt, da die zum Schema h passenden Chromosomen nur als Erwartungswert bekannt sind.

Das Schematheorem: Crossover

- **Crossover:**
  Die Auswirkungen des Crossover werden beschrieben durch:

$$N(h, t + \Delta t_s + \Delta t_c) = \underbrace{(1 - p_c) \cdot N(h, t + \Delta t_s)}_{A} + \underbrace{p_c \cdot N(h, t + \Delta t_s) \cdot (1 - p_{\text{loss}})}_{B} + C$$

p_c	Wahrscheinlichkeit eines Crossover
p_{loss}	Wahrscheinlichkeit, daß durch ein Ein-Punkt-Crossover die Passung eines Chromosoms zum Schema h verlorengeht
A	Erwartungswert der Anzahl Chromosomen, die zum Schema h passen und *nicht* am Crossover teilnehmen
B	Erwartungswert der Anzahl Chromosomen, die am Crossover teilnehmen und deren Passung zum Schema h dadurch nicht verlorengeht
C	Gewinne an Chromosomen, die zum Schema h passen, durch ... (siehe Übungsaufgabe)

Das Schematheorem: Crossover

Betrachtungen zur Wahrscheinlichkeit p_{loss}:

- **Beispiele:**

$$\begin{array}{rcccl} h = & \texttt{**0*}|\texttt{1*1*} & & \texttt{**0*1*1*} & = h \\ h \triangleright c_1 = & \texttt{0000}|\texttt{1111} & \rightarrow & \texttt{00000000} & = c_1' \not\triangleleft h \\ h \not\triangleright c_2 = & \texttt{1111}|\texttt{0000} & \rightarrow & \texttt{11111111} & = c_2' \not\triangleleft h \end{array}$$

$$\begin{array}{rcccl} h = & \texttt{**0*}|\texttt{1*1*} & & \texttt{**0*1*1*} & = h \\ h \triangleright c_1 = & \texttt{0000}|\texttt{1111} & \rightarrow & \texttt{00001010} & = c_1' \triangleleft h \\ h \triangleright c_2 = & \texttt{1101}|\texttt{1010} & \rightarrow & \texttt{11011111} & = c_2' \triangleleft h \end{array}$$

- **Folglich:**

$$p_{\text{loss}} \leq \underbrace{\frac{\text{dl}(h)}{L-1}}_{A = p_c^*(h)} \cdot \left(1 - \underbrace{\frac{N(h, t + \Delta t_s)}{\text{popsize}}}_{B}\right)$$

A	Wahrscheinlichkeit, daß der Schnittpunkt zwischen festgelegte Gene fällt
B	Wahrscheinlichkeit, daß das zweite Chromosom zum Schema h paßt

- **Frage:** Warum gilt nur $\leq$ und nicht $=$? (siehe Übungsaufgabe)

Das Schematheorem: Crossover

- Einsetzen des Ausdrucks für p_{loss} liefert:

$$\begin{aligned} & N(h, t + \Delta t_s + \Delta t_c) \\ \geq\ & (1 - p_c) \cdot N(h, t + \Delta t_s) \\ & + \quad p_c \cdot N(h, t + \Delta t_s) \cdot \left(1 - \frac{\text{dl}(h)}{L-1} \cdot \left(1 - \frac{N(h, t + \Delta t_s)}{\text{popsize}}\right)\right) \\ =\ & N(h, t + \Delta t_s) \left(1 - p_c + p_c \cdot \left(1 - \frac{\text{dl}(h)}{L-1} \cdot \left(1 - \frac{N(h, t + \Delta t_s)}{\text{popsize}}\right)\right)\right) \\ =\ & N(h, t + \Delta t_s) \cdot \left(1 - p_c \frac{\text{dl}(h)}{L-1} \cdot \left(1 - \frac{N(h, t + \Delta t_s)}{\text{popsize}}\right)\right) \\ \overset{(*)}{=}\ & N(h, t) \cdot f_{\text{rel}}(h) \cdot \text{popsize} \cdot \left(1 - p_c \frac{\text{dl}(h)}{L-1} \cdot (1 - N(h, t) \cdot f_{\text{rel}}(h))\right) \end{aligned}$$

- Im Schritt $(*)$ wurde zweimal die vorher abgeleitete Beziehung
  $N(h, t + \Delta t_s) = N(h, t) \cdot f_{\text{rel}}(h) \cdot \text{popsize}$ ausgenutzt.

Das Schematheorem: Mutation

- **Mutation:**
  Die Auswirkungen der Mutation werden durch die Ordnung beschrieben:

$$N(h, t+1) = N(h, t+\Delta t_s+\Delta t_c+\Delta t_m) = N(h, t+\Delta t_s+\Delta t_c) \cdot (1-p_m)^{\text{ord}(h)}$$

p_m	Mutationswahrscheinlichkeit für jedes Bit, d.h., jedes Bit wird mit Wahrscheinlichkeit p_m mutiert (umgedreht) und mit Wahrscheinlichkeit $(1 - p_m)$ unverändert gelassen.

Erläuterung: Damit die Passung nicht verlorengeht, darf keines der $\text{ord}(h)$ Gene verändert werden, die im Schema h festgelegt sind.

- **Alternative Modelle** sind möglich, z.B.:
  Jedes Chromosom wird genau einer Bitmutation unterworfen.

$$N(h, t + 1) = N(h, t + \Delta t_s + \Delta t_c + \Delta t_m) = N(h, t + \Delta t_s + \Delta t_c) \cdot \frac{\text{ord}(h)}{L}$$

$\frac{\text{ord}(h)}{L}$	Wahrscheinlichkeit, daß durch die Bitmutation ein im Schema h festgelegtes Gen verändert wird (gleichwahrscheinliche Auswahl des Bits).

Das Schematheorem

- Insgesamt (mit erstem Mutationsmodell):

$$\begin{aligned} N(h,t+1) &= f_{\text{rel}}(h) \cdot \text{popsize} \cdot \left(1 - p_c \frac{\text{dl}(h)}{L-1} \cdot (1 - N(h,t) \cdot f_{\text{rel}}(h))\right) \\ &\quad \cdot (1-p_m)^{\text{ord}(h)} \cdot N(h,t) \end{aligned}$$

- Einsetzen des Fitneßverhältnisses liefert das **Schematheorem**:

$$\begin{aligned} N(h,t+1) &= \frac{\overline{f_t(h)}}{\overline{f_t}} \left(1 - p_c \frac{\text{dl}(h)}{L-1} \left(1 - \frac{N(h,t)}{\text{popsize}} \cdot \frac{\overline{f_t(h)}}{\overline{f_t}}\right)\right) (1-p_m)^{\text{ord}(h)} \\ &\cdot \quad N(h,t) \end{aligned}$$

- **Interpretation:** Schemata mit
  - überdurchschnittlicher mittlerer Bewertung,
  - kurzer definierender Länge und
  - geringer Ordnung

  vermehren sich besonders stark (etwa exponentiell).

Das Schematheorem: Baustein-Hypothese

- Das Schematheorem besagt, daß der Suchraum besonders gut in Hyperebenen (Regionen) durchsucht wird, die Schemata mit hoher mittlerer Fitneß, kleiner definierender Länge und geringer Ordnung entsprechen.

  (Denn in diesen Regionen vermehren sich die Chromosomen am stärksten.)

- Solche Schemata heißen **Bausteine** (engl.: **building blocks**); die obige Aussage heißt daher auch **Baustein-Hypothese**.

- **Beachte:** Die obige Form der Bausteinhypothese gilt nur für Bitfolgen, fitneßproportionale Selektion, Standardmutation und Ein-Punkt-Crossover.

  Bei Verwendung z.B. anderer genetischer Operatoren werden Bausteine u.U. durch andere Eigenschaften charakterisiert (siehe Übungsaufgabe).

  (Die hohe mittlere Fitneß ist jedoch stets eine Eigenschaft, da jedes Selektionsverfahren Chromosomen mit hoher Fitneß bevorzugt, wenn auch unterschiedlich stark und nicht immer in direkter Relation zum Fitneßwert.)

Das Schematheorem: Kritik

- Das Schematheorem gilt in Strenge nur bei unendlicher Populationsgröße.

  (Sonst kann es Abweichungen von den betrachteten Erwartungswerten geben, die nicht vernachlässigbar sind.)

  Diese Annahme kann natürlich in der Praxis nie erfüllt werden. Es kommt daher immer zu Abweichungen vom idealen Verhalten (*stochastische Drift*).

- Es wird implizit angenommen, daß es kaum Wechselwirkungen zwischen den Genen gibt (geringe *Epistasie*), also die Fitneß von Chromosomen, die zu einem Schema passen, sehr ähnlich ist.

- Ein dritter, oft angeführter Einwand ist:

  Es wird implizit angenommen, daß interagierende Gene im Chromosom eng zusammen liegen, um möglichst kleine Bausteine (*building blocks*) zu bilden.

  Dieser Einwand betrifft jedoch eigentlich die Beschränkung auf das Ein-Punkt-Crossover und nicht den Ansatz an sich. Ihm kann durch andere Maße als die definierende Länge begegnet werden, die operationenspezifisch sind.

Genetische Algorithmen zur Verhaltenssimulation

Genetische Algorithmen zur Verhaltenssimulation

- Bisher: Verwendung genetischer Algorithmen um Optimierungsprobleme (numerische oder diskrete) zu lösen.
- Jetzt: Verwendung genetischer Algorithmen um Verhalten zu simulieren (Populationsdynamik) und Verhaltensstrategien zu finden.
- **Grundlage: Spieltheorie**
  - Dient der Analyse sozialer und wirtschaftlicher Situationen.
  - Modellierung von Handlungen als Spielzüge in einem festgelegten Rahmen.
  - Wichtigste theoretische Grundlage der Wirtschaftswissenschaften.
- **Allgemeiner Ansatz:**
  - Kodiere die Verhaltensstrategie eines Akteurs in einem Chromosom.
  - Lasse Akteure miteinander interagieren und bewerte ihren Erfolg.
  - Akteure vermehren sich oder sterben aus, je nach erzieltem Erfolg.

Das Gefangenendilemma

Das bekannteste Problem der Spieltheorie ist das **Gefangenendilemma** (engl.: prisoner's dilemma)

- Zwei Personen haben einen Banküberfall begangen und werden verhaftet.
- Die Beweise reichen jedoch nicht aus, um sie in einem Indizienprozeß wegen des Banküberfalls zu verurteilen.
- Die Beweise reichen jedoch aus, um sie wegen eines geringfügigeren Deliktes (z.B. unerlaubter Waffenbesitz) zu verurteilen (Strafmaß: 1 Jahr Gefängnis).
- Angebot des Staatsanwaltes: Kronzeugenregelung
  - Gesteht einer der beiden die Tat, wird er Kronzeuge und nicht verurteilt.
  - Der andere dagegen wird mit voller Härte bestraft (10 Jahre Gefängnis)
  - Problem: Gestehen beide, gilt die Kronzeugenregelung nicht.
    Da sie jedoch beide geständig sind, erhalten sie mildernde Umstände zugesprochen (Strafe: je 5 Jahre Gefängnis)

Das Gefangenendilemma

Zur Analyse des Gefangenendilemmas benutzt man eine **Auszahlungsmatrix**:

A \ B	schweigt	gesteht
schweigt	**−1** / **−1**	**−10** / **0**
gesteht	**0** / **−10**	**−5** / **−5**

- Kooperation (beide schweigen) ist insgesamt am günstigsten.
- **Aber:** Doppeltes Geständnis ist das **Nash-Gleichgewicht**.

  Nash-Gleichgewicht: Keine der beiden Seiten kann ihre Auszahlung erhöhen, wenn nur sie ihre Aktion ändert.

  Jede Auszahlungsmatrix hat mindestens ein Nash-Gleichgewicht [Nash 1950].

Das Gefangenendilemma

Allgemeine **Auszahlungsmatrix** des Gefangenendilemmas:

A \ B	cooperate	defect
cooperate	**R** / **R**	**S** / **T**
defect	**T** / **S**	**P** / **P**

R: Reward for mutual cooperation
T: Temptation to defect
P: Punishment for mutual defection
S: Sucker's payoff

- Die genauen Werte für **R**, **P**, **T** und **S** sind nicht wichtig.
- Es muß aber gelten $\mathbf{T} > \mathbf{R} > \mathbf{P} > \mathbf{S}$ und $2\,\mathbf{R} > \mathbf{T} + \mathbf{S}$.
  (Wenn die 2. Bedingung nicht erfüllt ist, ist wechselweises Ausbeuten besser.)

Das Gefangenendilemma

- Viele Alltagssituation kann man mit dem Gefangenendilemma beschreiben.
- **Aber:** Obwohl der doppelte Defekt das Nash-Gleichgewicht ist, beobachten wir auch anderes (kooperatives) Verhalten.
- Fragestellung (nach [Axelrod 1980]):
  Unter welchen Bedingungen entsteht Kooperation in einer Welt von Egoisten ohne zentrale Autorität?
- Antwort von [Hobbes 1651] (Leviathan):
  - **Gar nicht!**
    Ehe staatliche Ordnung existierte, wurde der Naturzustand dominiert von egoistischen Individuen, die so rücksichtslos gegeneinander wetteiferten, daß das Leben „solitary, poor, nasty, brutish, and short“ war.
  - **Aber:** Auf internationaler Ebene gibt es *de facto* keine zentrale Autorität, aber dennoch (wirtschaftliche und politische) Kooperation von Staaten.

Das Gefangenendilemma

- **Ansatz** von [Axelrod 1980]:
  Betrachte das **iterierte Gefangenendilemma**.
  (Das Gefangenendilemma wird von zwei Spielern mehrfach hintereinander gespielt, wobei sie die vergangenen Züge des jeweils anderen Spielers kennen.)
- **Idee** dieses Ansatzes:
  - Wird das Gefangenendilemma nur *einmal* gespielt, ist es am günstigsten, das Nash-Gleichgewicht zu wählen.
  - Wird das Gefangenendilemma *mehrfach* gespielt, kann ein Spieler auf unkooperatives Verhalten des anderen reagieren.
    (Möglichkeit der *Vergeltung* für erlittene Nachteile)
- Fragestellungen:
  Entsteht im iterierten Gefangenendilemma Kooperation?
  Was ist die beste Strategie im iterierten Gefangendilemma?

Das Gefangenendilemma

[Axelrod 1980] legte folgende **Auszahlungsmatrix** fest:

A \ B	cooperate	defect
cooperate	**3** / **3**	**0** / **5**
defect	**5** / **0**	**1** / **1**

(Satz von kleinsten nicht-negativen ganzen Zahlen, die die Bedingungen erfüllen.)

- Wissenschaftler verschiedener Disziplinen (Psychologie, Sozial- und Politikwissenschaften, Wirtschaftswissenschaften, Mathematik) wurden eingeladen, Programme zu schreiben, die das iterierte Gefangenendilemma spielen.
- Ein Programm kann sich die eigenen und die gegnerischen Züge merken.

Das Gefangenendilemma: Turniere

Zur Beantwortung der genannten Fragen führte Axelrod zwei Turniere durch:

- **1. Turnier**:
  - 14 Programme plus ein Zufallsspieler (Fortran)
  - Rundenturnier mit 200 Spielen je Paarung
  - Sieger: A. Rapoport mit Tit-for-Tat (Wie du mir, so ich dir.)
- Die Programme und Ergebnisse des ersten Turniers wurden veröffentlicht, dann wurde zu einem zweiten Turnier eingeladen.
  Idee: Ergebnisanalyse ermöglicht es ggf., bessere Programme zu schreiben.
- **2. Turnier**:
  - 62 Programme plus ein Zufallsspieler (Fortran und Basic)
  - Rundenturnier mit 200 Spielen je Paarung
  - Sieger: A. Rapoport mit Tit-for-Tat (Wie du mir, so ich dir.)

Das Gefangenendilemma: Tit-for-Tat

- Die Spielstrategie von **Tit-for-Tat** ist *sehr* einfach:
  - Kooperiere im ersten Spiel (spiele C).
  - Mache in allen folgenden Spielen den Zug des Gegners aus dem direkt vorangehenden Spiel.
- **Beachte**: Reines Tit-for-Tat ist nicht unbedingt die beste Strategie, wenn gegen *einzelne* andere Strategien gespielt wird.
  - Nur wenn es in einer Population Individuen gibt, mit denen Tit-for-Tat kooperieren kann, schneidet es insgesamt sehr gut ab.
  - **Problem** von Tit-for-Tat: Es ist **anfällig für Fehler**. Spielt Tit-for-Tat gegen Tit-for-Tat und spielt einer der beiden Spieler „aus Versehen“ Defekt, so kommt es zu wechselseitigen Vergeltungsschlägen.
- Eine wichtige Alternative ist **Tit-for-Two-Tat**: Schlage erst nach zweimaligem Defekt des Gegners zurück.

Das Gefangenendilemma: Genetischer Algorithmus

Kodierung der Spielstrategien: [Axelrod 1987]

- Betrachte alle möglichen Spielverläufe der Länge 3 ($2^6 = 64$ Möglichkeiten).
- Speichere für jeden Spielverlauf den im nächsten Spiel auszuführenden Zug (C — cooperate oder D — defect, kodiert in einem Bit):

		1. Spiel	2. Spiel	3. Spiel	
1. Bit:	Antwort auf	(C,C),	(C,C),	(C,C):	C
2. Bit:	Antwort auf	(C,C),	(C,C),	(C,D):	D
3. Bit:	Antwort auf	(C,C),	(C,C),	(D,C):	C
⋮		⋮		⋮	⋮
64. Bit:	Antwort auf	(D,D),	(D,D),	(D,D):	D

(Erstes Element jedes Paares: eigener Zug, zweites Element: Zug des Gegners)

- Zusätzlich: 6 Bit zur Kodierung des Spielverlaufs vor dem ersten Zug.
  → jedes Chromosom hat 70 binäre Gene (jeweils C oder D)

Genetischer Algorithmus: Ablauf

- Initialisierung einer Anfangspopulation mit zufälligen Bitfolgen (70 Bit).
- Aus der aktuellen Population werden Paare von Individuen zufällig ausgewählt. Sie spielen 200-mal das Gefangendilemma gegeneinander.

  Für die ersten drei Spiele wird (ein Teil des) im Chromosom abgespeicherten Anfangsspielverlaufs benutzt, um den Zug zu bestimmen. (Die fehlende/zu kurze Historie wird ersetzt/aufgefüllt.)
- Jedes Individuum spielt gegen die gleiche Anzahl von Gegnern. (aus Rechenzeitgründen — 1987! — kein volles Rundenturnier).
- Auswahl von Individuen für die nächste Generation:

überdurchschnittliches Ergebnis	$(x \geq \mu + \sigma)$:	2 Nachkommen
durchschnittliches Ergebnis	$(\mu - \sigma < x < \mu + \sigma)$:	1 Nachkomme
unterdurchschnittliches Ergebnis	$(\mu - \sigma \geq x)$:	0 Nachkommen

- Genetische Operatoren: Standardmutation, Ein-Punkt-Crossover

Genetischer Algorithmus: Ergebnis

- Die sich ergebenden Strategien sind **Tit-for-Tat** sehr ähnlich.
- [Axelrod 1987] identifizierte die folgenden allgemeinen Muster:
  - **Don't rock the boat.** Kooperiere nach drei Kooperationen. (C,C), (C,C), (C,C) → C
  - **Be provokable.** Spiele Defekt nach plötzlichen Defekt des Gegners. (C,C), (C,C), (C,D) → D
  - **Accept an apology.** Kooperiere, nach wechselseitiger Ausbeutung. (C,C), (C,D), (D,C) → C
  - **Forget.** (Sei nicht nachtragend.) Kooperiere, nachdem Kooperation nach einer Ausbeutung wiederhergestellt wurde (auch ohne Vergeltung). (C,C), (C,D), (C,C) → C
  - **Accept a rut.** (rut: *fig.* ausgefahrenes Gleis, alter Trott) Spiele Defekt nach dreimaligem Defekt des Gegners. (D,D), (D,D), (D,D) → D

Das Gefangenendilemma: Erweiterungen

Das Gefangenendilemma läßt sich in verschiedener Weise erweitern, um es realistischer zu machen und weitere Situationen zu erfassen:

- In der Praxis sind die Auswirkungen von Handlungen nicht immer perfekt beobachtbar. Daher: Nicht der genaue Zug des Gegners, sondern nur Wahrscheinlichkeiten sind bekannt.
- Oft sind mehr als zwei Akteure beteiligt: Mehr-Personen-Gefangenendilemma

Ebenso lassen sich die Beschreibungen der Strategien erweitern:

- Berücksichtigung längerer Spielverläufe (mehr als drei Spiele).
- Hinzunahme einer Zufallskomponente für die Wahl des Spielzugs: Wahrscheinlichkeiten für die Wahl von C und D statt eines festen Zuges.
- Beschreibung der Spielstrategie durch Moore-Automaten oder allgemeine Programme, die dann in einem genetischen Algorithmus verändert werden.

Genetische Programmierung

Genetische Programmierung

- **Bisher:** Darstellung von Lösungskandidaten / Spielstrategien durch Chromosomen fester Länge (Vektoren von Genen).
- **Jetzt:** Darstellung durch Funktionsausdrücke oder Programme.

  → **Genetische Programmierung**

  → komplexere Chromosomen variabler Länge
- Formale Grundlage: Grammatik zur Beschreibung der Sprache
- Festlegung zweier Mengen:
  - $\mathcal{F}$ — Menge der Funktionssymbole und Operatoren
  - $\mathcal{T}$ — Menge der Terminalsymbole (Konstanten und Variablen)
- Die Mengen $\mathcal{F}$ und $\mathcal{T}$ sind problemspezifisch.
  Sie sollten nicht zu groß sein (Beschränkung des Suchraums) und doch reichhaltig genug, um eine Problemlösung zu ermöglichen.

Genetische Programmierung

- **Beispiel 1**: Erlernen einer Booleschen Funktion
  - $\mathcal{F} = \{\text{and}, \text{or}, \text{not}, \text{if} \ldots \text{then} \ldots \text{else} \ldots, \ldots\}$
  - $\mathcal{T} = \{x_1, \ldots, x_m, 1, 0\}$ bzw. $\mathcal{T} = \{x_1, \ldots, x_m, t, f\}$
- **Beispiel 2:** Symbolische Regression

  (Regression: Bestimmung einer Ausgleichsfunktion zu gegebenen Daten unter Minimierung der Fehlerquadratsumme — *Methode der kleinsten Quadrate*)
  - $\mathcal{F} = \{+, -, *, /, \sqrt{\ }, \sin, \cos, \log, \exp, \ldots\}$
  - $\mathcal{T} = \{x_1, \ldots, x_m\} \cup \mathbb{R}$
- **Beachte:** Alle Chromosomen müssen auswertbar sein.
  Daher: Ggf. Vervollständigung des Definitionsbereichs einer Funktion, z.B.
  - $\forall x \leq 0: \quad \log(x) = 0$ oder = kleinste darstellbare Fließkommazahl
  - $\tan\left(\frac{\pi}{2}\right) = 0$ oder = größte darstellbare Fließkommazahl

Genetische Programmierung

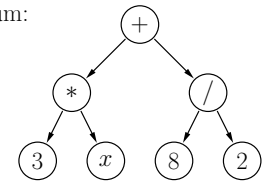
- Die Chromosomen sind nun Ausdrücke, die aus Elementen aus $\mathcal{C} = \mathcal{F} \cup \mathcal{T}$ zusammengesetzt sind (ggf. werden noch Klammern hinzugefügt).
- Allerdings: Beschränkung auf „wohlgeformte“ symbolische Ausdrücke.
  Übliche **rekursive Definition** (Präfixnotation):
  - Konstanten- und Variablensymbole sind symbolische Ausdrücke.
  - Sind $t_1, \ldots, t_n$ symbolische Ausdrücke und ist $f \in \mathcal{F}$ ein (n-stelliges) Funktionssymbol, so ist $(f\ t_1\ \ldots\ t_n)$ ein symbolischer Ausdruck.
  - Keine anderen Zeichenfolgen sind symbolische Ausdrücke.
- **Beispiele** zu dieser Definition:
  - „$(+\ (*\ 3\ x)\ (/\ 8\ 2))$“ ist ein symbolischer Ausdruck.
    Lisp- bzw. Scheme-artige Schreibweise, Bedeutung: $3 \cdot x + \frac{8}{2}$.
  - „$2\ 7\ *\ (\ 3\ /$“ ist kein symbolischer Ausdruck.

Genetische Programmierung

- Für die Implementierung der genetischen Programmierung ist es günstig, symbolische Ausdrücke durch sogenannte **Parse-Bäume** darzustellen.
  (Parse-Bäume werden in einem Parser z.B. eines Compilers verwendet, um arithmetische Ausdrücke darzustellen und anschließend zu optimieren.)

symbolischer Ausdruck: $(+\ (*\ 3\ x)\ (/\ 8\ 2))$

Parse-Baum:



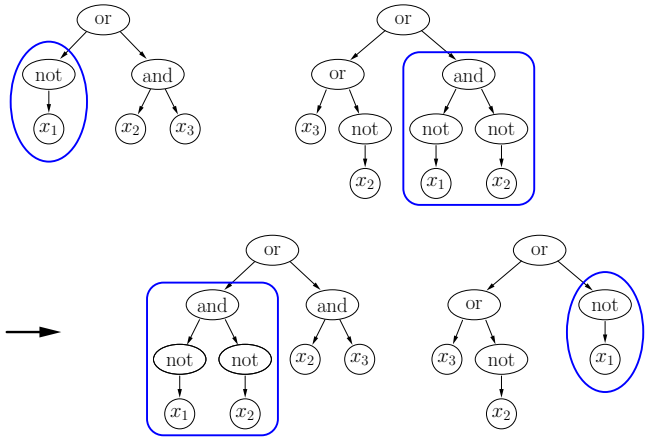
- In Lisp / Scheme werden Ausdrücke durch (verschachtelte) Listen dargestellt:
  - Das erste Element dieser Liste ist das Funktionssymbol bzw. der Operator.
  - Die folgenden Elemente sind die Argumente bzw. Operanden.

Genetische Programmierung: Ablauf

- Erzeugen einer **Anfangspopulation** zufälliger symbolischer Ausdrücke.
  Parameter des Erzeugungsprozesses:
  - maximale Verschachtelungstiefe (maximale Baumhöhe)
  - Wahrscheinlichkeit für die Wahl eines Terminalsymbols
- **Bewertung** der Ausdrücke durch Berechnung der Fitneß.
  - Erlernen Boolescher Funktionen:
    Anteil korrekter Ausgaben für alle Eingaben bzw. in einer Stichprobe.
  - Symbolische Regression:
    Summe der Fehlerquadrate über die gegebenen Meßpunkte.
    eindimensional: Daten (x_i, y_i), $i = 1, \ldots, n$, Fitneß $f(c) = \sum_{i=1}^{n} (c(x_i) - y_i)^2$
- **Selektion** mit einem der besprochenen Verfahren.
- Anwendung **genetischer Operatoren**, meist nur Crossover.

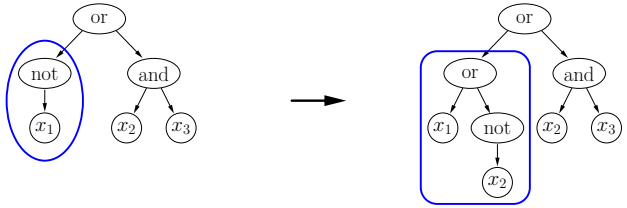
Genetische Programmierung: Crossover

- Crossover besteht im Austauschen zweier Teilausdrücke (Teilbäume)



Genetische Programmierung

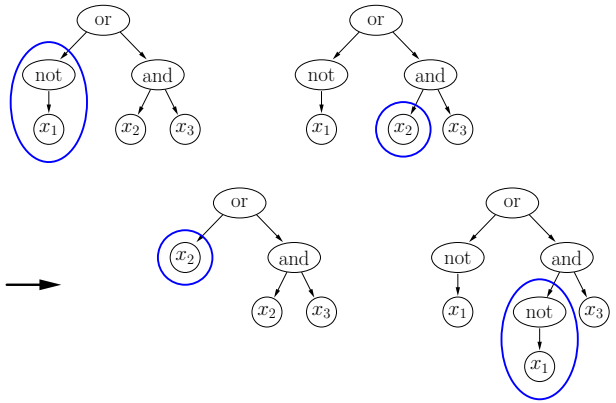
- Mutation besteht im Ersetzen eines Teilausdrucks (Teilbaums) durch einen zufällig erzeugten neuen:



- Es sollten möglichst nur kleine Teilbäume ersetzt werden.
- Meist wird nur Crossover und keine Mutation verwendet.
- Es sollte dann aber darauf geachtet werden, daß die Population groß genug ist, um einen hinreichend großen Vorrat an „genetischem Material“ zur Verfügung zu haben, aus dem neue Lösungskandidaten rekombiniert werden können.

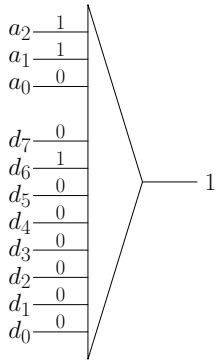
Genetische Programmierung

- **Beachte:** Crossover ist mächtiger als für Vektoren, denn ein Crossover identischer Eltern kann zu neuen Individuen führen.



Genetische Programmierung: 11-Multiplexer

Beispiel: Erlernen eines Booleschen 11-Multiplexers [Koza 1992]



- Multiplexer mit 8 Daten- und 3 Adreßleitungen
  (Der Zustand der Adreßleitungen gibt die durchzuschaltende Datenleitung an.)
- $2^{11} = 2048$ mögliche Eingaben mit je einer zugehörigen Ausgabe
- Festlegung der Symbolmengen:
  - $\mathcal{T} = \{a_0, a_1, a_2, d_0, \ldots, d_7\}$
  - $\mathcal{F} = \{\text{and}, \text{or}, \text{not}, \text{if}\}$
- Fitneßfunktion: $f(s) = 2048 - \sum_{i=1}^{2048} e_i$, wobei e_i der Fehler für die i-te Eingabe ist.

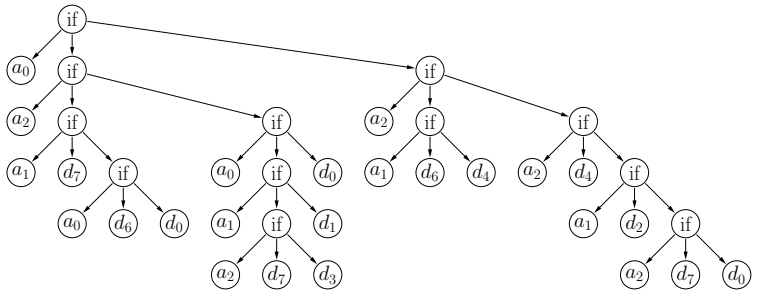
Genetische Programmierung: 11-Multiplexer

Beispiel: Erlernen eines Booleschen 11-Multiplexers [Koza 1992]

- Populationsgröße popsize = 4000
- Anfangstiefe der Parse-Bäume: 6, maximale Tiefe: 17
- Die Fitneßwerte in der Anfangspopulation zwischen 768 und 1280, die mittlere Fitneß liegt bei 1063.
  (Der Erwartungswert liegt bei 1024, da bei zufälliger Ausgabe im Durchschnitt die Hälfte der Ausgaben richtig sind.)
- 23 Ausdrücke haben eine Fitneß von 1280, einer davon entspricht einem 3-Multiplexer: (if a_0 d_1 d_2)
- Glücksradauswahl (fitneßproportionale Selektion)
- 90% (3600) der Individuen werden Crossover unterworfen, die restlichen unverändert übernommen.

Genetische Programmierung: 11-Multiplexer

- Nach nur 9 Generationen ist folgende Lösung gefunden (Fitneß 2048):

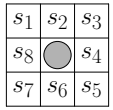


- Dieses Ergebnis ist erstaunlich einfach und daher nicht plausibel. Es steht zu vermuten, daß die Ergebnisgüte der genetischen Programmierung übertrieben wird, was sich z.B. durch Auswahl des besten von vielen Läufen erreichen läßt.

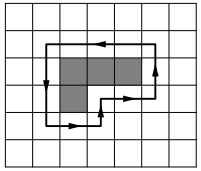
Genetische Programmierung: Stimulus-Response-Agent

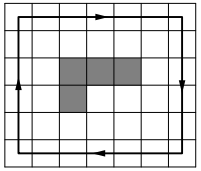
Beispiel: Erlernen eines Robotersteuerprogramms [Nilsson 1998]

- Betrachte einen Stimulus-Response-Agenten in einer Gitterwelt:



  - 8 Sensoren $s_1, \ldots, s_8$ liefern Zustand der Nachbarfelder
  - 4 Aktionen: go east, go north, go west, go south
  - Direkte Berechnung der Aktion aus den Sensoreingaben, kein Gedächtnis
- **Aufgabe:** Umlaufe ein im Raum stehendes Hindernis oder laufe die Begrenzung des Raumes ab.





Genetische Programmierung: Stimulus-Response-Agent

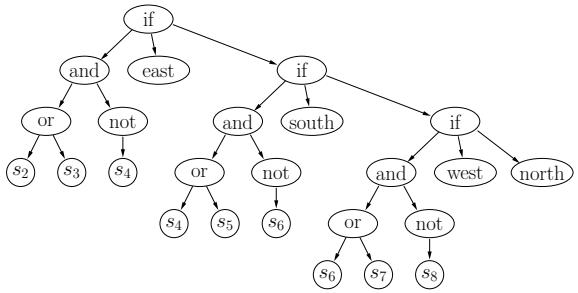
- Symbolmengen:
  - $\mathcal{T} = \{s_1, \ldots, s_8, \text{east}, \text{north}, \text{west}, \text{south}, 0, 1\}$
  - $\mathcal{F} = \{\text{and}, \text{or}, \text{not}, \text{if}\}$
- Vervollständigung der Funktionen, z.B. durch

$$(\text{and } x\ y) = \begin{cases} \text{false}, & \text{falls } x = \text{false}, \\ y, & \text{sonst.} \end{cases}$$

  (Beachte: So kann auch eine logische Operation eine Aktion liefern.)
- Populationsgröße popsize = 5000, Turnierauswahl mit Turniergröße 5
- Aufbau der Nachfolgepopulation
  - 10% (500) Lösungskandidaten werden unverändert übernommen.
  - 90% (4500) Lösungskandidaten werden durch Crossover erzeugt.
  - <1% der Lösungskandidaten werden mutiert.
- 10 Generationen (ohne Anfangspopulation) werden berechnet.

Genetische Programmierung: Stimulus-Response-Agent

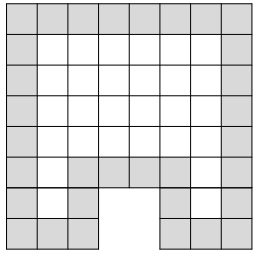
- **Optimale, von Hand konstruierte Lösung:**



- Es ist höchst unwahrscheinlich, daß genau diese Lösung gefunden wird.
- Um die Chromosomen einfach zu halten, ist es u.U. sinnvoll, einen Strafterm zu berechnen, der die Komplexität des Ausdrucks mißt.

Genetische Programmierung: Stimulus-Response-Agent

- Bewertung einer Kandidatenlösung anhand eines Testraumes:



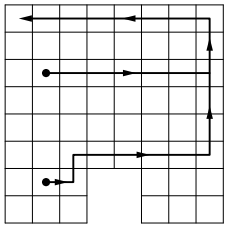
  - Ein perfekt arbeitendes Steuerprogramm läßt den Agenten die grau gezeichneten Felder ablaufen.
  - Das Startfeld wird zufällig gewählt.
  - Ist eine Aktion nicht ausführbar oder wird statt einer Aktion ein Wahrheitswert geliefert, so wird die Ausführung des Steuerprogramms abgebrochen.

- Ein durch ein Chromosom gesteuerter Agent wird auf 10 zufällige Startfelder gesetzt und seine Bewegung verfolgt.
- Die Zahl der insgesamt besuchten Randfelder (grau unterlegt) ist die Fitneß. (maximale Fitneß: $10 \cdot 32 = 320$)

Genetische Programmierung: Stimulus-Response-Agent

```
(and (not (not (if  (if  (not s1)
                         (if  s4 north east)
                         (if  west 0 south))
                    (or  (if  s1 s3 s8) (not s7))
                    (not (not north)))))
     (if  (or  (not (and (if  s7 north s3)
                         (and south 1)))
               (or  (or  (not s6) (or s4 s4))
                    (and (if  west s3 s5)
                         (if  1 s4 s4))))
          (or  (not (and (not s3)
                         (if  east s6 s2)))
               (or  (not (if  s1 east s6))
                    (and (if  s8 s7 1)
                         (or  s7 s1))))
          (or  (not (if  (or  s2 s8)
                         (or  0 s5)
                         (or  1 east)))
               (or  (and (or  1 s3)
                         (and s1 east))
                    (if  (not west)
                         (and west east)
                         (if  1 north s8))))))
```

Bestes Individuum in Generation 0:

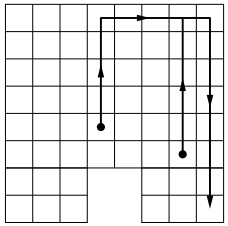


(Bewegung von zwei Startpunkten aus)

Genetische Programmierung: Stimulus-Response-Agent

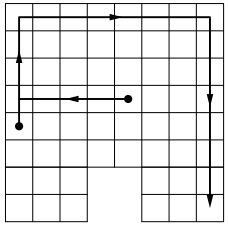
Bestes Individuum in Generation 2:

```
(not (and (if  s3
               (if  s5 south east)
               north)
          (and not s4)))
```



Bestes Individuum in Generation 6:

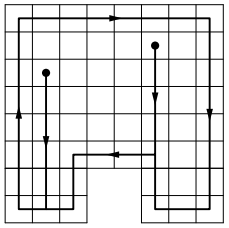
```
(if  (and (not s4)
          (if  s4 s6 s3))
     (or  (if  1  s4 south)
          (if  north east s3))
     (if  (or  (and 0 north)
               (and s4 (if  s4
                            (if  s5 south east)
                            north)))
          (and s4 (not (if  s6 s7 s4)))
          (or  (or  (and s1 east) west) s1)))
```



Genetische Programmierung: Stimulus-Response-Agent

Bestes Individuum in Generation 10:

```
(if  (if  (if  s5 0 s3)
          (or  s5 east)
          (if  (or  (and s4 0)
                    s7)
               (or  s7 0)
               (and (not (not (and s6 s5)))
                    s5)))
     (if  s8
          (or north
              (not (not s6)))
          west)
     (not (not (not (and (if  (not south)
                              s5
                              s8)
                         (not s2))))))
```



- Dieses Ergebnis ist erstaunlich einfach und daher nicht plausibel. Es steht zu vermuten, daß die Ergebnisgüte der genetischen Programmierung übertrieben wird, was sich z.B. durch Auswahl des besten von vielen Läufen erreichen läßt.

Genetische Programmierung: Stimulus-Response-Agent

Mit dem Programm von der Vorlesungsseite gefundene Lösung:

```
(or  (if  s3
          (if  (not (and s4 s4))
               east
               (not south))
          (not south))
     (if  (if  north
               (not (and (not s7)
                         (or  west south)))
               (and (not (not north))
                    s4))
          (if  (or  (or  (if  (or  east s4)
                              (and s8   s1)
                              (or  s7   s1))
                         (not (not south)))
                    (not (or  s5
                              (if  north
                                   north
                                   north))))
               (if  (or  (if  (and west south)
                              (and s3   east)
                              (if  south
                                   s8
                                   s4))
                         (not (or  s3 s8)))
                    west
                    (or  (or  (or  north s1)
                              (not north))
                         (and s4
                              (and s1 s4))))
               (and s4
                    (not (and north s8))))
     (or  (if  (if  north
                    s6
                    s8)
               (or  west  west)
               (or  south s7))
          (if  (and s6    s4)
               s3
               s7))))
```


Genetische Programmierung: Erweiterungen

- **Umformung (Editing)**
  - Dient der Vereinfachung der Chromosomen.
  - Ein Chromosom oder Chromosomenteil kann auf komplizierte Weise eine (sehr) einfache Funktion ausdrücken.
  - Durch eine Umformung, die logische und/oder arithmetische Äquivalenzen ausnutzt, wird der Ausdruck vereinfacht.
- **Kapselung (Encapsulation) / automatisch definierte Funktionen**
  - Potentiell gute Teilausdrücke sollten vor Zerstörung durch Crossover und Mutation geschützt werden.
  - Für einen Teilausdruck (eines guten Chromosoms) wird eine neue Funktion definiert, das sie bezeichnende Symbol ggf. der Menge $\mathcal{F}$ hinzugefügt.
  - Die Zahl der Argumente der neuen Funktion ist gleich der Zahl der (verschiedenen) Blätter des Teilbaums.

Evolutionsstrategien

Evolutionsstrategien

- Bisher : Behandlung beliebiger (auch diskreter) Optimierungsprobleme
  Jetzt : Beschränkung auf **numerische Optimierung**

  Gegeben: Funktion $f : \mathbb{R}^n \to \mathbb{R}$
  Gesucht : Minimum oder Maximum von f

  Folglich : Die Chromosomen sind **Vektoren reeller Zahlen**.
- **Mutation** besteht in der Addition eines normalverteilten **Zufallsvektors** $\vec{r}$:

  Jedes Element r_i des Zufallsvektors ist die Realisierung einer normalverteilten Zufallsvariable mit
  - Erwartungswert 0 (unabhängig vom Elementindex i) und
  - Varianz σ_i^2 bzw. Standardabweichung σ_i.

  Die Varianz σ_i^2 kann abhängig oder unabhängig vom Elementindex i und abhängig oder unabhängig von der Generationsnummer t sein.
- Auf **Crossover** (Rekombination zweier Vektoren) wird oft verzichtet.

Evolutionsstrategien: Selektion

- **Strenges Eliteprinzip:**
  Nur die besten Individuen kommen in die nächste Generation.
- Bezeichnungen: μ – Anzahl der Individuen in der Elterngeneration
  λ – Anzahl der (durch Mutation) erzeugten Nachkommen
- **Zwei prinzipielle Selektionsstrategien:**
  - **+-Strategie** (Plus-Strategie, $(\mu + \lambda)$-Strategie)
    Aus den $(\mu + \lambda)$ Individuen der Eltergeneration und der erzeugten Nachkommen werden die besten μ Chromosomen ausgewählt.
    (Bei dieser Strategie gilt meist $\lambda < \mu$.)
  - **,-Strategie** (Komma-Strategie, (μ, λ)-Strategie)
    Es werden $\lambda > \mu$ Nachkommen erzeugt,
    aus denen die besten μ Chromosomen ausgewählt werden.
    (Die Chromosomen der Elterngeneration gehen auf jeden Fall verloren.)

Evolutionsstrategien: Selektion

- **Beispiel:** Spezialfall der (1+1)-Strategie
  - Anfangs„population“: $\vec{x}_0$ (zufällig erzeugter Vektor reeller Zahlen)
  - Erzeugen der nächsten Generation:
    Erzeuge einen reellen Zufallsvektor $\vec{r}_t$ und berechne $\vec{x}_t^* = \vec{x}_t + \vec{r}_t$.
    Setze dann
    $$\vec{x}_{t+1} = \begin{cases} \vec{x}_t^*, & \text{falls } f(\vec{x}_t^*) \geq f(\vec{x}), \\ \vec{x}_t, & \text{sonst.} \end{cases}$$
  - Erzeuge weitere Generationen bis ein Abbruchkriterium erfüllt ist.
- Dies entspricht offenbar dem am Anfang der Vorlesung besprochenen **Zufallsaufstieg**.
- Die allgemeine +-Strategie kann folglich als paralleler Zufallsaufstieg gesehen werden, der gleichzeitig an mehreren Orten des Suchraums durchgeführt wird, wobei stets die erfolgversprechendsten μ Wege verfolgt werden.

Evolutionsstrategien

- Zur Erinnerung (aus der Liste der Prinzipien der organismischen Evolution):
  Evolutionsstrategische Prinzipien
  Optimiert werden nicht nur die Organismen, sondern auch die Mechanismen der Evolution: Vermehrungs- und Sterberaten, Lebensdauern, Anfälligkeit gegenüber Mutationen, Mutationsschrittweiten, Evolutionsgeschwindigkeit etc.
- Hier: Anpassung der Varianz des Zufallsvektors (Mutationsschrittweite)
  - Geringe Varianz $\rightarrow$ kleine Änderungen an den Chromsomen
    $\rightarrow$ lokale Suche (Ausbeutung)
  - Hohe Varianz $\rightarrow$ große Änderungen an den Chromsomen
    $\rightarrow$ globale Suche (Durchforstung)
- Weitere Möglichkeiten, Parameter des Evolutionsprozesses anzupassen:
  - Wahl der Zahl der zu ändernden Gene (Vektorelemente).
  - Wahl der Zahl λ der zu erzeugenden Nachkommen.

Evolutionsstrategien: Varianzanpassung

Globale Varianzanpassung (chromosomenunabhängige Varianz)

- Idee: Wähle σ^2 bzw. σ so, daß die mittlere Konvergenzrate möglichst hoch ist.
- Ansatz von [Rechenberg 1973]: Bestimme optimale Varianz σ für
  - $f_1(x_1, \ldots, x_n) = a + bx_1$ und
  - $f_2(x_1, \ldots, x_n) = \sum_{i=1}^n x_i^2$,

  indem die Wahrscheinlichkeiten für eine erfolgreiche (d.h. verbessernde) Mutation bestimmt werden. Diese sind
  - für f_1: $p_1 \approx 0.184$ und
  - für f_2: $p_2 \approx 0.270$.
- Aus diesem Ergebnis wurde heuristisch die **$\frac{1}{5}$-Erfolgsregel** abgeleitet:
  In der +-Strategie ist die Mutationsschrittweite richtig,
  wenn etwa $\frac{1}{5}$ der Nachkommen besser sind als die Eltern.

Evolutionsstrategien: Varianzanpassung

Globale Varianzanpassung (chromosomenunabhängige Varianz)

- Anpassung der Varianz σ^2 auf der Grundlage der $\frac{1}{5}$-Erfolgsregel:
  - Sind mehr als $\frac{1}{5}$ der Nachkommen besser als die Eltern, vergrößere die Varianz/Standardabweichung.
    $$\sigma^{(\text{neu})} = \sigma^{(\text{alt})} \cdot c_{\text{inc}}, \qquad c_{\text{inc}} \approx 1.22.$$
  - Sind weniger als $\frac{1}{5}$ der Nachkommen besser als die Eltern, verkleinere die Varianz/Standardabweichung.
    $$\sigma^{(\text{neu})} = \sigma^{(\text{alt})} \cdot c_{\text{dec}}, \qquad c_{\text{dec}} = \frac{1}{c_{\text{inc}}} \approx 0.83.$$
- Bei größeren Populationen ist die $\frac{1}{5}$-Erfolgsregel z.T. zu optimistisch.
- Man kann analog zum **simulierten Ausglühen** auch eine Funktion definieren, mit der die Varianz im Laufe der Generationen verkleinert wird.

Evolutionsstrategien: Varianzanpassung

Lokale Varianzanpassung (chromosomenspezifische Varianz)

- Die Varianz/Standardabweichung wird in die Chromosonen aufgenommen:
  - Eine Varianz für alle Vektorelemente oder
  - eine individuelle Varianz für jedes Vektorelement (doppelte Vektorlänge)
- **Beachte:** Die zusätzlichen Vektorelemente für die Varianz(en) haben *keinen direkten Einfluß* auf die Fitneß eines Chromosoms.
- **Erwartung:** Chromosomen mit „schlechten“ Varianzen, d.h.
  - zu klein: Chromosomen entwickeln sich nicht schnell genug weiter oder
  - zu groß: Chromosomen entfernen sich zu weit von ihren Eltern,

  erzeugen vergleichsweise mehr „schlechte“ Nachkommen.
  Dadurch sterben ihre Gene (und damit auch ihre Varianzen) leichter aus.

Evolutionsstrategien: Varianzanpassung

Lokale Varianzanpassung (chromosomenspezifische Varianz)

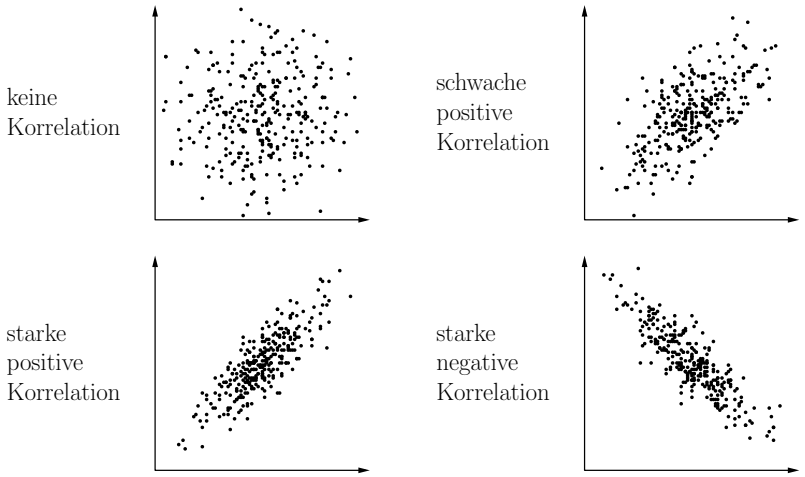
- Die elementspezifischen Mutationsschrittweiten (Standardabweichungen) werden nach dem folgenden Schema mutiert:
  $$\sigma_i^{(\text{neu})} = \sigma_i^{(\text{alt})} \cdot \exp(r_1 \cdot N(0,1) + r_2 \cdot N_i(0,1)).$$
  $N(0,1)$: einmal je Chromosom zu bestimmende normalverteilte Zufallszahl
  $N_i(0,1)$: für jedes Element/Gen zu bestimmende normalverteilte Zufallszahl
- Empfohlene Werte für die Parameter r_1 und r_2 sind [Bäck und Schwefel 1993]
  $$r_1 = \frac{1}{\sqrt{2n}}, \qquad r_2 = \frac{1}{\sqrt{2\sqrt{n}}},$$
  wobei n die Anzahl der Vektorelemente ist, oder [Nissen 1997]
  $$r_1 = 0.1, \qquad r_2 = 0.2.$$
- Oft wird eine untere Schranke für die Mutationsschrittweiten festgelegt

Evolutionsstrategien: Varianzanpassung

Erweiterungen der lokalen Varianzanpassung

- In der Standardform der lokalen Varianzanpassung sind die Varianzen der verschiedenen Vektorelemente unabhängig voneinander.
  (Formal: Die Kovarianzmatrix ist eine Diagonalmatrix.)
- Sollen die Variationen eines Chromosoms bevorzugt in bestimmten Richtungen erzeugt werden, so kann dies mit Einzelvarianzen nur ausgedrückt werden, wenn diese Richtungen achsenparallel sind.
- **Beispiel:** Variationen von Chromosomen mit zwei Genen sollen bevorzugt in Richtung der Hauptdiagonale, d.h. in Richtung $(1,1)$, erzeugt werden.
  Dies kann mit Einzelvarianzen nicht beschrieben werden.
- **Lösung:** Benutze eine Kovarianzmatrix mit hoher Kovarianz, z.B.
  $$\mathbf{\Sigma} = \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1 \end{pmatrix}.$$

Kovarianz und Korrelation



Cholesky-Zerlegung

- Anschaulich: **Berechne ein Analogon der Standardabweichung**.
- Sei $\mathbf{S}$ eine symmetrische, positiv definite Matrix (d.h. eine Kovarianzmatrix). Die Cholesky-Zerlegung dient dazu, eine „Quadratwurzel“ von $\mathbf{S}$ zu berechnen.
  - symmetrisch: $\forall 1 \leq i, j \leq m : s_{ij} = s_{ji}$
  - positiv definit: für alle m-dimensionalen Vektoren $\vec{v} \neq \vec{0}$ gilt $\vec{v}^\top \mathbf{S}\, \vec{v} > 0$
- Formal: Berechne eine linke/untere Dreiecksmatrix $\mathbf{L}$, so daß $\mathbf{L}\mathbf{L}^\top = \mathbf{S}$. ($\mathbf{L}^\top$ ist die Transponierte der Matrix $\mathbf{L}$.)

$$l_{ii} = \left(s_{ii} - \sum_{k=1}^{i-1} l_{ik}^2 \right)^{\frac{1}{2}}$$

$$l_{ji} = \frac{1}{l_{ii}} \left(s_{ij} - \sum_{k=1}^{i-1} l_{ik} l_{jk} \right), \qquad j = i+1, i+2, \ldots, m.$$

Cholesky-Zerlegung

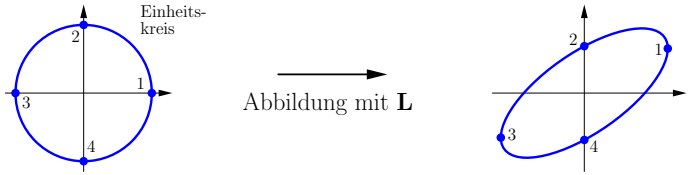
Spezialfall: Zwei Dimensionen

- Kovarianzmatrix

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix}$$

- Cholesky-Zerlegung

$$\mathbf{L} = \begin{pmatrix} \sigma_x & 0 \\ \frac{\sigma_{xy}}{\sigma_x} & \frac{1}{\sigma_x}\sqrt{\sigma_x^2\sigma_y^2 - \sigma_{xy}^2} \end{pmatrix}$$



Eigenwertzerlegung

- Liefert auch ein **Analogon der Standardabweichung**.
- Rechenaufwendiger als die Cholesky-Zerlegung.
- Sei $\mathbf{S}$ eine symmetrische, positiv definite Matrix (d.h. eine Kovarianzmatrix).
  - $\mathbf{S}$ kann geschrieben werden als

$$\mathbf{S} = \mathbf{R}\, \mathrm{diag}(\lambda_1, \ldots, \lambda_m)\, \mathbf{R}^{-1},$$

    wobei die λ_j, $j = 1, \ldots, m$, die Eigenwerte von $\mathbf{S}$ und die Spalten von $\mathbf{R}$ die (normierten) Eigenvektoren von $\mathbf{S}$ sind.
  - Die Eigenwerte λ_j, $j = 1, \ldots, m$, von $\mathbf{S}$ sind alle positiv und die Eigenvektoren von $\mathbf{S}$ sind orthonormal ($\rightarrow \mathbf{R}^{-1} = \mathbf{R}^\top$).
- Folglich kann $\mathbf{S}$ geschrieben werden als $\mathbf{S} = \mathbf{T}\,\mathbf{T}^\top$ mit

$$\mathbf{T} = \mathbf{R}\, \mathrm{diag}\left(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_m} \right)$$

Eigenwertzerlegung

Spezialfall: Zwei Dimensionen

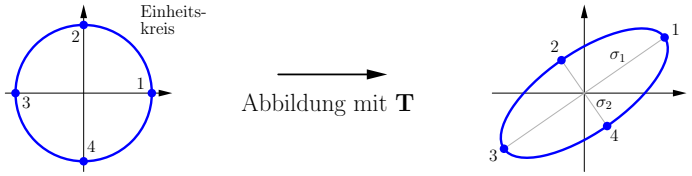
- Kovarianzmatrix

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix}$$

- Eigenwertzerlegung

$$\mathbf{T} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix},$$

$$s = \sin\phi, c = \cos\phi, \phi = \tfrac{1}{2}\arctan\tfrac{\sigma_{xy}}{\sigma_y^2 - \sigma_x^2},$$
$$\sigma_1 = \sqrt{c^2\sigma_x^2 + s^2\sigma_y^2 - 2sc\sigma_{xy}},$$
$$\sigma_2 = \sqrt{s^2\sigma_x^2 + c^2\sigma_y^2 + 2sc\sigma_{xy}}.$$



Evolutionsstrategien: Varianzanpassung

- **Allgemein:** Korrelierte Mutation wird durch n Varianzen und $\frac{n(n-1)}{2}$ Rotationswinkel beschrieben.
- Es wird dann die Kovarianzmatrix

$$\mathbf{\Sigma} = \left(\prod_{i=1}^{n}\prod_{k=i+1}^{n} R_{ik}(\varphi_{ik})\right) \operatorname{diag}(\sigma_1^2, \ldots, \sigma_n^2) \left(\prod_{i=1}^{n}\prod_{k=i+1}^{n} R_{ik}(\varphi_{ik})\right)^{-1}$$

verwendet, wobei

$$R_{ik}(\varphi_{ik}) = \begin{pmatrix} 1 & & & & & & & & & & \\ & \ddots & & & & & & & & & \\ & & 1 & & & & & & & & \\ & & & \cos\varphi & & & & \sin\varphi & & & \\ & & & & 1 & & & & & & \\ & & & & & \ddots & & & & & \\ & & & & & & 1 & & & & \\ & & & -\sin\varphi & & & & \cos\varphi & & & \\ & & & & & & & & 1 & & \\ & & & & & & & & & \ddots & \\ & & & & & & & & & & 1 \end{pmatrix}.$$

Evolutionsstrategien: Varianzanpassung

- Die Kovarianzmatrix $\mathbf{\Sigma}$ ist i.a. chromosomenspezifisch.
  (Ein Chromosom hat dann $n + \frac{n(n+1)}{2}$ Gene.)
- Die Mutation der Kovarianzen wird auf den Rotationswinkeln ausgeführt, nicht direkt auf den Einträgen der Matrix. Als Mutationsregel wird

$$\varphi_{ik}^{(\text{neu})} = \varphi_{ik}^{(\text{alt})} + r \cdot N(0,1)$$

  mit $r \approx 0.0873$ ($\approx 5^o$) verwendet.
  $N(0,1)$ ist eine in jedem Schritt neu zu wählende normalverteilte Zufallszahl.
- **Nachteile** der korrelierten Mutation:
  - Es müssen deutlich mehr Parameter angepaßt werden.
  - Die Varianzen und Rotationswinkel haben keinen direkten Einfluß auf die Fitneßfunktion; ihre Anpassung geschieht eher „beiläufig“.
    Es ist daher fraglich, ob die Anpassung der Winkel der Veränderung der eigentlich zu optimierenden Parameter schnell genug folgen kann.

Evolutionsstrategien: Crossover/Rekombination

- **Zufällige Auswahl von Komponenten** aus den Eltern:

$$\begin{matrix} (\mathbf{x_1}, x_2, x_3, \ldots, \mathbf{x_{n-1}}, x_n) \\ (y_1, \mathbf{y_2}, \mathbf{y_3}, \ldots, y_{n-1}, \mathbf{y_n}) \end{matrix} \quad \Rightarrow \quad (x_1, y_2, y_3, \ldots, x_{n-1}, y_n)$$

  Dieser Ansatz entspricht dem **uniformen Crossover**.
  Im Prinzip ist auch die Anwendung von 1-, 2- oder n-Punkt-Crossover möglich.
- **Mittelung** (blending, intermediäre Rekombination):

$$\begin{matrix} (x_1, \ldots, x_n) \\ (y_1, \ldots, y_n) \end{matrix} \quad \Rightarrow \quad \tfrac{1}{2}(x_1 + y_1, \ldots, x_n + y_n)$$

- **Achtung:** Bei Mittelung besteht die Gefahr des **Jenkins Nightmare**.
  (Zur Erinnerung: Das *Jenkins Nightmare* besteht im völligen Verschwinden jeglicher Verschiedenheit in einer Population. Es wird durch die Mittelung begünstigt, da die Gene dann einem mittleren Wert zustreben.)

Evolutionsstrategien: Plus- versus Komma-Strategien

- Offensichtlicher **Vorteil** der +-Strategie:
  - Es treten, wegen des strengen Eliteprinzips, nur Verbesserungen auf.
- **Nachteile:**
  - Gefahr des Hängenbleibens in lokalen Minima.
  - Für eine $(\mu + \lambda)$-Evolutionsstrategie mit

    $\frac{\mu}{\lambda} \geq$ „beste Wahrscheinlichkeit für eine erfolgreiche Mutation“ $\quad (\approx \frac{1}{5})$

    haben die Chromosomen einen Selektionsvorteil, die ihre Varianz σ^2 möglichst klein halten, da nicht genügend große Mutationen durchgeführt werden, um eine „echte“ Verbesserung zu erreichen („Beinahe-Stagnation“).

    Übliche Wahl des Verhältnisses von μ und λ: etwa 1:7.
- Wenn über mehrere Generationen keine Verbesserungen aufgetreten sind, wird oft von der +-Strategie vorübergehend auf die ,-Strategie umgeschaltet, um die Diversität in der Population wieder zu erhöhen.

Mehrkriterienoptimierung

Mehrkriterienoptimierung

- In vielen Alltagsproblemen wird nicht eine einzelne Größe optimiert, sondern **verschiedene Ziele** sollen zu möglichst hohem Grad erreicht werden.
- **Beispiel:** Beim Autokauf wünscht man sich
  - einen niedrigen Preis,
  - einen geringen Kraftstoffverbrauch,
  - möglichst viel Komfort wie elektrische Fensterheber oder eine Klimaanlage.
- Die verschiedenen, zu erreichenden Ziele sind oft nicht unabhängig, sondern **gegensätzlich**: Sie können nicht alle gleichzeitig voll erreicht werden.
- **Beispiel:** Autokauf
  - Für viele Ausstattungsmerkmale ist ein Aufpreis zu zahlen.
  - Die Wahl z.B. einer Klimaanlage oder einfach ein geräumigeres Auto bedingen oft einen etwas leistungsfähigeren Motor und damit einen höheren Preis und Kraftstoffverbrauch.

Mehrkriterienoptimierung

- **Formale Beschreibung:** Es sind k Kriterien gegeben, denen jeweils eine zu optimierende Zielfunktion zugeordnet ist:

$$f_i : S \to \mathbb{R}, \qquad i = 1, \ldots, k$$

- **Einfachster Lösungsansatz:** Fasse die k Zielfunktionen zu einer Gesamtzielfunktion zusammen, z.B. durch Bildung einer gewichteten Summe

$$f(s) = \sum_{i=1}^{k} w_i f_i(s).$$

- **Wahl der Gewichte:**
  - *Vorzeichen:* Soll die Gesamtzielfunktion zu maximieren sein, so müssen die Vorzeichen der Gewichte w_i der Zielfunktionen f_i, die zu maximieren sind, positiv, die der Gewichte der übrigen Zielfunktionen negativ sein.
  - *Absolutwert:* Mit den Absolutwerten der Gewichte wird die relative Wichtigkeit der Kriterien ausgedrückt (Schwankungsbreite berücksichtigen!).

Mehrkriterienoptimierung

- **Probleme** des Ansatzes mit einer gewichteten Summe der Zielfunktionen:
  - Man muß bereits vor Beginn der Suche festlegen, welche relative Wichtigkeit die verschiedenen Kriterien haben.
  - Es ist nicht immer einfach, die Gewichte so zu wählen, daß die Präferenzen zwischen den Kriterien angemessen wiedergegeben werden.
- Die Probleme, die mit einer Linearkombination der Zielfunktionen auftreten, sind jedoch noch viel fundamentaler:
  - Allgemein stellt sich das Problem der **Aggregation von Präferenzordnungen**.
  - Dieses Problem tritt auch bei Personenwahlen auf. (Die Kandidatenpräferenzen der Wähler müssen zusammengefaßt werden.)
  - **Arrowsches Paradoxon** [Arrow 1951]: Es gibt keine Wahlfunktion, die alle wünschenswerten Eigenschaften hat.

Mehrkriterienoptimierung

- Die Arrowschen Unmöglichkeitssätze [Arrow 1951] lassen sich im Prinzip durch Verwendung **skalierter Präferenzordnungen** umgehen.
- **Aber:** Die Skalierung der Präferenzordnung ist ein weiterer Freiheitsgrad. Es ist u.U. noch schwieriger, eine passende Skalierung zu finden, als die Gewichte einer Linearkombination angemessen zu bestimmen.
- **Alternativer Ansatz:** Versuche, alle bzw. möglichst viele **Pareto-optimale** Lösungen zu finden.
- Ein Element s des Suchraums S heißt **Pareto-optimal** bezüglich der Zielfunktionen f_i, $i = 1, \ldots, k$, wenn es *kein* Element $s' \in S$ gibt, für das gilt

$$\begin{array}{lll} \forall i, 1 \leq i \leq k: & f_i(s') \geq f_i(s) & \text{und} \\ \exists i, 1 \leq i \leq k: & f_i(s') > f_i(s). & \end{array}$$

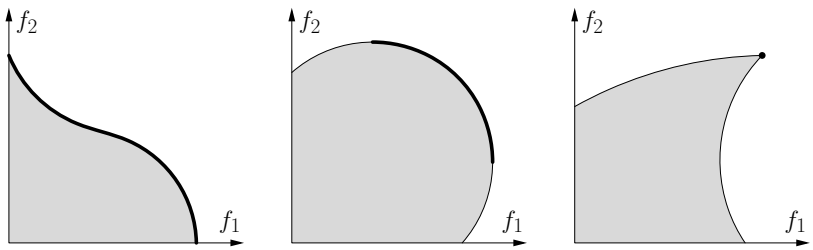
- **Anschaulich:** Der Wert keiner Zielfunktion kann verbessert werden, ohne den Wert einer anderen zu verschlechtern.

Mehrkriterienoptimierung

- Ausführlichere Definition des Begriffs „Pareto-optimal“:
  - Ein Element $s_1 \in S$ **dominiert** ein Element $s_2 \in S$, wenn gilt
    $$\forall i, 1 \leq i \leq k: \quad f_i(s_1) \geq f_i(s_2).$$
  - Ein Element $s_1 \in S$ **dominiert** ein Element $s_2 \in S$ **echt**, wenn s_1 s_2 dominiert und außerdem gilt
    $$\exists i, 1 \leq i \leq k: \quad f_i(s_1) > f_i(s_2).$$
  - Ein Element $s_1 \in S$ heißt **Pareto-optimal**, wenn es von keinem Element $s_2 \in S$ echt dominiert wird.
- **Vorteile** der Suche nach Pareto-optimalen Lösungen:
  - Die Zielfunktionen müssen nicht zusammengefaßt werden, die Bestimmung von Gewichten entfällt.
  - Die Suche muß auch für verschiedene Präferenzen nur einmal durchgeführt werden, da erst anschließend aus den gefundenen Lösungen gewählt wird.

Mehrkriterienoptimierung

Veranschaulichung Pareto-optimaler Lösungen

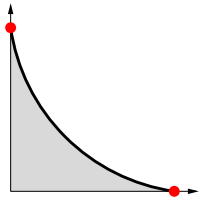


- Alle Punkte des Suchraums liegen im grau gezeichneten Bereich.
- Pareto-optimale Lösungen liegen auf dem fett gezeichneten Teil des Randes.
- Man beachte, daß je nach Lage der Lösungskandidaten die Pareto-optimale Lösung auch eindeutig bestimmt sein kann (siehe rechtes Diagramm).

Mehrkriterienoptimierung mit genetischen Algorithmen

- **Einfachster Ansatz:** Verwende eine gewichtete Summe der einzelnen Zielfunktionen als Fitneßfunktion. (Dies hat die erwähnten Nachteile.)
  - Bemerkung: Dieser Ansatz führt zu einer Pareto-optimalen Lösung, aber eben einer durch die Gewichtungen ausgezeichneten.
- **Naheliegende Alternative**: Gegeben seien k Kriterien, denen die Zielfunktionen f_i, $1 \ldots, k$ zugeordnet sind.

  $\forall i, 1, \ldots, k$: Wähle $\frac{\text{popsize}}{k}$ Individuen auf Grundlage der Fitneßfunktion f_i.
  - **Vorteil**: einfach, geringer Rechenaufwand.
  - **Nachteil:** Lösungen, die alle Kriterien recht gut, aber keines maximal erfüllen, haben einen deutlichen Selektionsnachteil.
  - **Folge:** Suche konzentriert sich auf Randlösungen.



Mehrkriterienoptimierung mit genetischen Algorithmen

- **Besserer Ansatz:** Nutze Dominanzbegriff zur Selektion.
- Aufbau einer **Rangskala** der Individuen einer Population:
  - Finde alle nicht dominierten Lösungskandidaten der Population.
  - Ordne diesen Lösungskandidaten den höchsten Rang zu und entferne sie aus der Population.
  - Wiederhole das Bestimmen und Entfernen der nicht dominierten Lösungskandidaten für die weiteren Ränge, bis die Population leer ist.
- Führe mit Hilfe der Rangskala eine **Rangauswahl** durch.
- Meist wird der Ansatz mit **Nischentechniken** kombiniert (um zwischen Individuen mit gleichem Rang zu unterscheiden).
- **Alternative: Turnierauswahl**, wobei der Turniersieger über den Dominanzbegriff und ggf. Nischentechniken bestimmt wird.

Parallelisierung genetischer Algorithmen

Parallelisierung genetischer Algorithmen

- Genetische Algorithmen sind recht **teure Optimierungsverfahren**, da oft
  - mit einer großen Population (einige tausend bis einige zehntausend Individuen)
  - mit einer großen Zahl an Generationen (einige hundert)

  gearbeitet werden muß, um eine hinreichende Lösungsgüte zu erreichen.
- Dieser Nachteil wird zwar durch eine oft etwas höhere Lösungsgüte im Vergleich zu anderen Verfahren wettgemacht, trotzdem kann die Laufzeit eines genetischen Algorithmus unangenehm lang sein.
- Möglicher Lösungsansatz: **Parallelisierung**, d.h. die Verteilung der notwendigen Operationen auf mehrere Prozessoren.
- Fragen:
  - **Welche Schritte kann man parallelisieren?**
  - **Welche vorteilhaften Techniken kann man bei der Parallelisierung zusätzlich anwenden?**

Parallelisierung: Was kann man parallelisieren?

- **Erzeugen der Anfangspopulation**

  Meist problemlos parallelisierbar, da i.a. die Chromosomen der Anfangspopulation zufällig und unabhängig voneinander erzeugt werden.

  Der Versuch, Duplikate zu vermeiden, kann die Parallelisierung behindern.

  Die Parallelisierung dieses Schrittes hat eher geringe Bedeutung, da eine Anfangspopulation nur einmal erzeugt wird.

- **Bewertung der Chromosomen**

  Problemlos parallelisierbar, da die Chromosomen i.a. unabhängig voneinander bewertet werden (Fitneß hängt nur vom Chromosom selbst ab).

  Auch beim Gefangenendilemma können Paarungen parallel bearbeitet werden.

- **Berechnung der (relativen) Fitneßwerte**

  Zur Berechnung der relativen Fitneß oder einer Rangordnung der Chromosomen müssen die Bewertungen zusammengeführt werden.

Parallelisierung: Was kann man parallelisieren?

- **Selektion**

  Ob sich der Auswahlschritt parallelisieren läßt, hängt sehr stark vom verwendeten Selektionsverfahren ab:

  - **Erwartungswertmodell** und **Elitismus**

    Erfordern beide eine globale Betrachtung der Population und sind daher nur schwer parallelisierbar.

  - **Glücksrad-** und **Rangauswahl**

    Nachdem die relativen Fitneßwerte bzw. die Ränge bestimmt sind (was schwer parallelisierbar ist), ist die Auswahl selbst leicht parallelisierbar.

  - **Turnierauswahl**

    Ideal für die Parallelisierung geeignet, besonders bei kleinen Turniergrößen, da keine globale Information bestimmt werden muß, sondern sich der Vergleich der Fitneßwerte auf die Individuen des Turniers beschränkt.

Parallelisierung: Was kann man parallelisieren?

- **Anwendung genetischer Operatoren**

  Leicht zu parallelisieren, da jeweils nur ein (Mutation) oder zwei Chromosomen (Crossover) betroffen sind. Zusammen mit einer Turnierauswahl kann ein Steady-State genetischer Algorithmus daher sehr gut parallelisiert werden.

- **Abbruchbedingung**

  Der einfache Test, ob eine bestimmte Generationenzahl erreicht ist, bereitet bei einer Parallelisierung keine Probleme.

  Abbruchkriterien wie

  - das beste Individuum der Population hat eine bestimmte Mindestgüte oder
  - über eine bestimmte Anzahl von Generationen hat sich das beste Individuum nicht/kaum verbessert

  sind dagegen für eine Parallelisierung weniger geeignet, da sie eine globale Betrachtung der Population erforden.

Parallelisierung: Inselmodell und Migration

- Auch wenn z.B. ein schwer zu parallelisierendes Selektionsverfahren verwendet wird, kann man eine Parallelisierung erreichen, indem mehrere unabhängige Populationen parallel berechnet werden.

  Jede Population wird als eine Insel bewohnend angesehen, daher **Inselmodell**.

- Das reine Inselmodell ist äquivalent zu einer mehrfachen seriellen Ausführung des gleichen genetischen Algorithmus. Es liefert meist etwas schlechtere Ergebnisse als ein einzelner Lauf mit entsprechend größerer Population.

- Zwischen den Inselpopulationen können zu festgelegten Zeitpunkten (*nicht* in jeder Generation) Individuen ausgetauscht werden.

  Man spricht in diesem Fall von **Migration** (Wanderung).

- Es findet normalerweise keine direkte Rekombination von Chromosomen statt, die von verschiedenen Inseln stammen. Erst nach einer Migration wird genetische Information von einer Insel mit der einer anderen Insel kombiniert.

Parallelisierung: Inselmodell und Migration

- **Steuerung der Migration zwischen Inseln**
  - **Zufallsmodell**
    Die beiden Inseln, zwischen denen Individuen ausgetauscht werden sollen, werden zufällig bestimmt. Beliebige Inseln können Individuen austauschen.
  - **Netzwerkmodell**
    Die Inseln werden in einem Graphen angeordnet. Individuen können zwischen den Inseln nur entlang der Kanten des Graphen wandern. Die Kanten, über die Individuen ausgetauscht werden sollen, werden zufällig bestimmt.
- **Wettbewerb zwischen den Inseln**
  - Die genetischen Algorithmen, die auf den Inseln angewandt werden, unterscheiden sich (in Verfahren und/oder Parametern).
  - Die Populationsgröße einer Insel wird entsprechend ihrer durchschnittlichen Fitneß der Individuen erhöht oder erniedrigt. Es gibt jedoch eine Mindestpopulationsgröße, die nicht unterschritten werden darf.

Parallelisierung: Zellulare genetische Algorithmen

(auch: **"isolation by distance"**)

- Die Prozessoren werden in einem (rechtwinkligen) Gitter angeordnet.
  Das Gitter bedeckt gewöhnlich die Oberfläche eines Torus.
- Selektion und Crossover werden auf im Gitter benachbarte (durch Kanten verbundene), Mutation auf einzelne Prozessoren beschränkt.
- Beispiel: Jeder Prozessor verwaltet ein Chromosom.
  - **Selektion:** Ein Prozessor wählt das beste Chromosom seiner (vier) Nachbarprozessoren oder eines dieser Chromosomen zufällig nach ihrer Fitneß.
  - **Crossover:** Der Prozessor führt Crossover mit dem gewählten und dem eigenen Chromosom durch oder mutiert sein Chromosom. Er behält den besseren der beiden Nachkommen bzw. von Elter und Kind.
- Es bilden sich Gruppen benachbarter Prozessoren, die ähnliche Chromosomen verwalten. Dies mildert die oft zerstörende Wirkung des Crossover.

Parallelisierung: Mühlenbeins Ansatz

- **Kombination** von genetischen Algorithmen **mit Zufallsaufstieg**:
  Jedes Individuum führt lokalen Zufallsaufstieg durch, d.h.
  - bei einer vorteilhaften Mutation wird der Elter ersetzt,
  - bei einer nachteiligen Mutation bleibt der Elter erhalten.

  Dieser Zufallsaufstieg kann leicht parallelisiert werden.
- Individuen suchen sich einen Crossover-Partner in ihrer Nachbarschaft.
  (Dazu wird eine Definition des Abstandes zweier Individuen benötigt — vergleiche die Nischentechniken, z.B. *power law sharing*)
- Die Nachkommen (Crossover-Produkte) führen lokalen Zufallsaufstieg durch.
- Die Individuen der nächsten Generation werden nach dem **„lokalen" Eliteprinzip** ausgewählt, d.h., die beiden besten Individuen unter den Eltern und den optimierten Nachkommen werden übernommen.

Anwendungsbeispiele

1. Flugroutenplanung: ROGENA

Problemstellung: Flugroutenplanung

- Flugzeuge bewegen sich im Luftraum normalerweise
  - auf Standardrouten zwischen den Flughäfen,
  - mit einem vorgeschriebenen Mindestabstand auf der gleichen Route,
  - abhängig von ihrer Flugrichtung auf unterschiedlichen Höhen.
- **Vorteile** dieser Lösung:
  - einfache Regeln/Vorschriften für den Flugverkehr,
  - leichte Kontrolle der Flugrouten durch die Fluglotsen,
  - große Sicherheit im Flugverkehr (i.w. Kollisionsvermeidung).
- **Nachteile** dieser Lösung:
  - nur ein relativ kleiner Teil des Luftraums wird genutzt,
  - die Flugzeugdichte in der Nähe von Flughäfen ist relativ eng begrenzt.

→ Diese Lösung ist dem steigenden Flugverkehr nicht mehr angemessen.

Luftraum in der Umgebung von Frankfurt

picture not available in online version

Lösungsidee: Fluglotsenunterstützung

- Der Luftraum zwischen den Standardrouten muß genutzt werden, um für zukünftige Luftverkehrssteigerungen gewappnet zu sein.
- **Problem** dieses Ansatzes:
  - Sich freier bewegende Flugzeuge sind schwerer zu koordinieren.
  - Dadurch steigt die Arbeitsbelastung für die Fluglotsen.
- **Aufgabenstellung:** Entwicklung eines Unterstützungswerkzeugs, das die Konstruktion sicherer und effizienter Routen zwischen Eintritts- und Austrittspunkt in dem vom Lotsen kontrollierten Teil des Luftraumes übernimmt.
- **Vorgegebene Rahmenbedingungen:**
  - Eintrittszeitpunkt, Ein- und Austrittsort im kontrollierten Bereich,
  - Flugeigenschaften der kontrollierten Flugzeuge,
  - Bewegung weiterer Flugzeuge (Kollisionsvermeidung),
  - Sperrgebiete (Bebauung, militärische Nutzung, schlechtes Wetter).

Lösungsansatz mit genetischen Algorithmen

- Der Suchraum für mögliche Flugrouten ist sehr groß; eine analytische Konstruktionsmethode ist schwer zu finden.
  → Suche mit genetischen Algorithmen
- **ROGENA** (free ROuting with GENetic Algorithms) [Gerdes 1994, 1995]
- Die notwendigen Daten und Berechnungsformeln für die Beschreibung der Flugeigenschaften und des Flugverhaltens von Flugzeugen wurden der BADA-Datenbank der EUROCONTROL entnommen [Byrne 1995].
- Es wurde ein Ausschnitt von 200 × 200 nautischen Meilen (NM) mit einer Höhe von 0 bis 10000 Fuß (ft) aus dem Luftraum betrachtet.
- Mindestabstände zwischen Flugzeugen: standardmäßig 5 NM, beim Endanflug zwischen 2.5 und 6 NM in Abhängigkeit vom Gewicht der beiden Flugzeuge.
- Der genetische Algorithmus beginnt erst bei Flugroutenkonflikten (Unterschreitung der Sicherheitsabstände) zu arbeiten.

ROGENA: Kodierung der Lösungskandidaten

- Eine Flugroute wird als Sequenz von Linienstücken dargestellt (vgl. Übungsaufgabe zum Finden eines Weges durch ein Gebiet).
- Die Chromosomen haben variable Länge (variable Anzahl von Genen).
- Jedes Gen stellt einen zu überfliegenden Punkt dar; zusätzlich wird die Überflug-geschwindigkeit angegeben:

	x	y	z	v
Gen 1	86.2	15.8	1.65	258
Gen 2	88.9	24.7	1.31	252
⋮	⋮	⋮	⋯	⋮
Gen k	105.0	98.0	0.00	120

- Es müssen Nebenbedingungen eingehalten werden, z.B.
  - monoton fallende Flugroute (da Landeanflüge modelliert werden),
  - maximale Beschleunigung/Verzögerung zwischen Punkten,
  - minimaler Winkel zwischen Linienstücken (Kurvenradius) etc.

ROGENA: Fitneßfunktion

- Die Fitneßfunktion berücksichtigt folgende Eigenschaften einer Flugroute:
  - einzuhaltender Sicherheitsabstand zu anderen Flugzeugen,
  - kein Durchfliegen von gesperrtem Luftraum,
  - Länge der Flugroute bis zur Landebahn,
  - Pünklichkeit des Fluges (planmäßige Ankunft),
  - möglichst geringe Abweichungen von der optimalen Sinkrate (berechnet nach BADA-Datenbank),
  - keine zu spitzen Winkel zwischen den Linienstücken, um Abweichungen von der tatsächlichen Flugbahn klein zu halten.
- Diese Eigenschaften gehen mit Gewichtungsfaktoren versehen in die Fitneßfunktion ein. Die Fitneßfunktion ist zu minimieren.
- Ein Benutzer kann die Gewichtungsfaktoren für die einzelnen Eigenschaften über Schieberegler beeinflußen.

ROGENA: Ablauf des genetischen Algorithmus

- **Ziel:** Erzeugen einer sicheren und effizienten Flugroute für ein neu in den kontrollierten Luftraum eintretendes Flugzeug.
- **Basis:** modifizierte Form eines genetischen Algorithmus (ein Chromosom wird entweder Crossover oder Mutation unterworfen)
- **Populationsgröße:** 60 Individuen/Chromosomen
- Initialisierung der **Anfangspopulation**: Durch wiederholte zufällige Veränderung der Standardflugroute.
- **Selektionsverfahren:** Im wesentlichen Glücksradauswahl, wobei allerdings Chromosomen, deren Fitneß über einem Schwellenwert liegt, nicht in die Berechnung der Selektionswahrscheinlichkeit eingehen (entspricht der Forderung einer Mindestgüte).

  Der Schwellenwert wird im Laufe der Generationen gesenkt, und zwar in Abhängigkeit vom Fitneßdurchschnitt der Population. (Kombination aus Glücksradauswahl und Sintflutalgorithmus)

ROGENA: Anwendung genetischer Operatoren

- 20 Chromosomen werden unverändert übernommen, darunter die 5 besten Chromosomen (Elitismus).
- 20 Chrosomen werden einem **2-Punkt-Crossover** unterworfen.
- 20 Chromosomen werden einem speziellen **Mutationsoperator** unterworfen:
  - entweder völlig zufällige Koordinatenänderung (globale Suche)
  - oder mittlere Veränderung in einen Punkt „in der Nähe“
  - oder kleine Veränderung in einen Punkt „in der Nähe“ (Zufallsaufstieg).
  - Zusätzlich Veränderung der Genanzahl (mit geringer Wahrscheinlichkeit). Der Lösch- oder Einfügeort im Chromosom wird zufällig bestimmt, der neue Punkt zufällig in der Nähe seiner Nachbarn initialisiert.
- **Reparaturmechanismen**: Eine Abfolge von Sink- und Steigvorgängen ist unökonomisch, ebenso eine Abfolge von Beschleunigungs- und Bremsvorgängen.

ROGENA: 2-Punkt-Crossover

picture not available in online version

Diese Form des Crossover wurde gewählt, um Teilstücke von Routen auszutauschen und auf diese Weise nützliche Teile verschiedener Routen zu kombinieren.

ROGENA: Benutzeroberfläche

picture not available in online version

ROGENA: Testläufe und Ergebnisse

- **Benutzeroberfläche:**
  - Originalroute in rot (direkte Verbindung von Eintrittsort und Zielpunkt)
  - beste aktuelle Route in grün
  - alternative Routen in grau
  - Höhendiagramm der Route in getrenntem Fenster
  - Links: Zeitleiter mit Landezeiten aller Flugzeuge
- Es wurden mehrere **Simulationsläufe** mit realen Daten durchgeführt (beschreiben Eintrittsort/-zeitpunkt und tatsächliche Flugbahn).
- Die von ROGENA erzeugten Flugrouten sind deutlich kürzer als die tatsächlichen, ohne daß dadurch Konflikte mit anderen Flugzeugen erzeugt wurden.
- Wirkungsvolle Unterstützung eines Fluglotsen bei der Koordinierung, die Verkürzung der Routen ermöglicht einen höheren Flugzeugdurchsatz.

Anwendungsbeispiele

2. Erlernen von Fuzzy-Reglern

Kurzeinführung Fuzzy-Theorie

- **Klassische Logik:** nur Wahrheitswerte *wahr* und *falsch*.
  Klassische Mengenlehre: entweder *ist Element* oder *ist nicht Element*.
- Die Zweiwertigkeit der klassischen Theorien ist oft nicht angemessen.

  Beispiel zur Illustration: **Sorites-Paradoxon** (griech. *sorites*: Haufen)
  - Eine Milliarde Sandkörner sind ein Sandhaufen. *(wahr)*
  - Wenn man von einem Sandhaufen ein Sandkorn entfernt, bleibt ein Sandhaufen übrig. *(wahr)*

  Es folgt daher:
  - 999 999 999 Sandkörner sind ein Sandhaufen. *(wahr)*

  Mehrfache Wiederholung des gleichen Schlusses liefert schließlich
  - 1 Sandkorn ist ein Sandhaufen. *(falsch!)*

  Bei welcher Anzahl Sandkörner ist der Schluß nicht wahrheitsbewahrend?

Kurzeinführung Fuzzy-Theorie

- Offenbar: Es gibt keine genau bestimmte Anzahl Sandkörner, bei der der Schluß auf die nächstkleinere Anzahl falsch ist.
- Problem: Begriffe der natürlichen Sprache (z.B. „Sandhaufen“, „kahlköpfig“, „warm“, „schnell“, „hoher Druck“, „leicht“ etc.) sind **vage**.
- Beachte: Vage Begriffe sind zwar *unexakt*, aber trotzdem nicht *unbrauchbar*.
  - Auch für vage Begriffe gibt es Situationen/Objekte, auf die sie *sicher anwendbar* sind und solche auf die sie *sicher nicht anwendbar* sind.
  - Dazwischen liegt eine **Penumbra** (lat. für *Halbschatten*) von Situationen, in denen es unklar ist, ob die Begriffe anwendbar sind, oder in denen sie nur mit Einschränkungen anwendbar sind („kleiner Sandhaufen“).
  - Die Fuzzy-Theorie versucht, diese Penumbra mathematisch zu modellieren („weicher Übergang“ zwischen *anwendbar* und *nicht anwendbar*).

Fuzzy-Logik

- Die **Fuzzy-Logik** ist eine Erweiterung der klassischen Logik um Zwischenwerte zwischen *wahr* und *falsch*.
- Als Wahrheitswert kann jeder Wert aus dem reellen Intervall $[0, 1]$ auftreten, wobei $0 \mathrel{\widehat{=}}$ *falsch* und $1 \mathrel{\widehat{=}}$ *wahr*.
- Folglich notwendig: **Erweiterung der logischen Operatoren**
  - Negation klassisch: $\neg a$, fuzzy: $\sim a$ Fuzzy-Negation
  - Konjunktion klassisch: $a \wedge b$, fuzzy: $\top(a, b)$ t-Norm
  - Disjunktion klassisch: $a \vee b$, fuzzy: $\bot(a, b)$ t-Konorm
- **Grundprinzipien** der Erweiterung:
  - Für die Extremwerte 0 und 1 sollen sich die Operationen genauso verhalten wie ihre klassischen Vorbilder (Rand-/Eckbedingungen).
  - Für die Zwischenwerte soll das Verhalten monoton sein.
  - Soweit möglich, sollen die Gesetze der klassischen Logik erhalten werden.

Fuzzy-Negationen

Eine **Fuzzy-Negation** ist eine Funktion $\sim\colon [0, 1] \to [0, 1]$, die die folgenden Bedingungen erfüllt:

- $\sim 0 = 1$ und $\sim 1 = 0$ (Randbedingungen)
- $\forall a, b \in [0, 1]: \quad a \leq b \;\Rightarrow\; \sim a \geq \sim b$ (Monotonie)

Gelten in der zweiten Bedingung statt $\leq$ und $\geq$ sogar die Beziehungen $<$ und $>$, so spricht man von einer *strikten* Negation.

Weitere Bedingungen, die manchmal gestellt werden, sind:

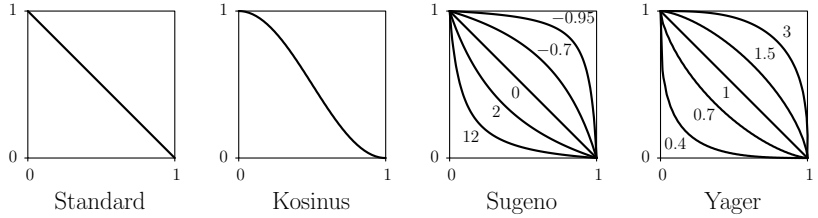
- $\sim$ ist eine stetige Funktion.
- $\sim$ ist *involutiv*, d.h. $\forall a \in [0, 1]: \quad \sim\sim a = a$.

Involutivität entspricht dem klassischen *Gesetz der Identität* $\neg\neg a = a$.

Die obigen Bedingungen legen die Fuzzy-Negation nicht eindeutig fest.

Fuzzy-Negationen

Standardnegation:	$\sim a$	$=$	$1 - a$
Schwellenwertnegation:	$\sim(a;\theta)$	$=$	$\begin{cases} 1, & \text{falls } x \leq \theta, \\ 0, & \text{sonst.} \end{cases}$
Kosinusnegation:	$\sim a$	$=$	$\frac{1}{2}(1 + \cos \pi a)$
Sugeno-Negation:	$\sim(a;\lambda)$	$=$	$\dfrac{1-a}{1+\lambda a}$
Yager-Negation:	$\sim(a;\lambda)$	$=$	$(1 - a^{\lambda})^{\frac{1}{\lambda}}$



t-Normen / Fuzzy-Konjunktionen

Eine **t-Norm** oder **Fuzzy-Konjunktion** ist eine Funktion $\top : [0,1]^2 \to [0,1]$, die die folgenden Bedingungen erfüllt:

- $\forall a \in [0,1]: \quad \top(a,1) = a$ (Randbedingung)
- $\forall a,b,c \in [0,1]: \quad b \leq c \;\Rightarrow\; \top(a,b) \leq \top(a,c)$ (Monotonie)
- $\forall a,b \in [0,1]: \quad \top(a,b) = \top(b,a)$ (Kommutativität)
- $\forall a,b,c \in [0,1]: \quad \top(a,\top(b,c)) = \top(\top(a,b),c)$ (Assoziativität)

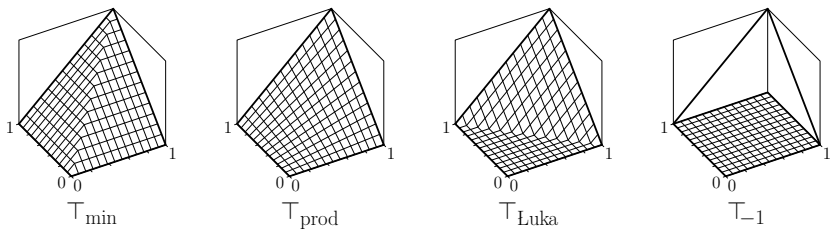
Weitere Bedingungen, die manchmal gestellt werden, sind:

- $\top$ ist eine stetige Funktion (Stetigkeit)
- $\forall a \in [0,1]: \quad \top(a,a) < a$ (Subidempotenz)
- $\forall a,b,c,d \in [0,1]: \quad a < b \wedge c < d \;\Rightarrow\; \top(a,b) < \top(c,d)$ (strikte Monotonie)

Die ersten beiden dieser Bedingungen (zusätzlich zu den ersten vier) definieren die Teilklasse der sogenannten *Archimedischen t-Normen*.

t-Normen / Fuzzy-Konjunktionen

Standardkonjunktion:	$\top_{\min}(a,b)$	$= \min\{a,b\}$
Algebraisches Product:	$\top_{\text{prod}}(a,b)$	$= a \cdot b$
Łukasiewicz:	$\top_{\text{Łuka}}(a,b)$	$= \max\{0, a+b-1\}$
Drastisches Product:	$\top_{-1}(a,b)$	$= \begin{cases} a, & \text{if } b = 1, \\ b, & \text{if } a = 1, \\ 0, & \text{otherwise.} \end{cases}$



t-Konormen / Fuzzy-Disjunktionen

Eine **t-Konorm** oder **Fuzzy-Disjunktion** ist eine Funktion $\perp : [0,1]^2 \to [0,1]$, die die folgenden Bedingungen erfüllt:

- $\forall a \in [0,1]: \quad \perp(a,0) = a$ (Randbedingung)
- $\forall a,b,c \in [0,1]: \quad b \leq c \;\Rightarrow\; \perp(a,b) \leq \perp(a,c)$ (Monotonie)
- $\forall a,b \in [0,1]: \quad \perp(a,b) = \perp(b,a)$ (Kommutativität)
- $\forall a,b,c \in [0,1]: \quad \perp(a,\perp(b,c)) = \perp(\perp(a,b),c)$ (Assoziativität)

Weitere Bedingungen, die manchmal gestellt werden, sind:

- $\perp$ ist eine stetige Funktion (Stetigkeit)
- $\forall a \in [0,1]: \quad \perp(a,a) > a$ (Superidempotenz)
- $\forall a,b,c,d \in [0,1]: \quad a < b \wedge c < d \;\Rightarrow\; \perp(a,b) < \perp(c,d)$ (strikte Monotonie)

Die ersten beiden dieser Bedingungen (zusätzlich zu den ersten vier) definieren die Teilklasse der sogenannten *Archimedischen t-Konormen*.

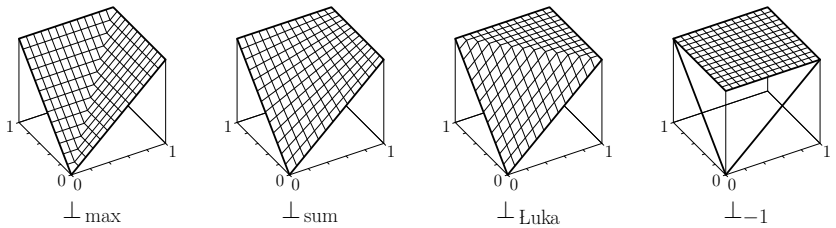
t-Konormen / Fuzzy-Disjunktionen

Standarddisjunktion: $\perp_{\max}(a,b) = \max\{a,b\}$

Algebraische Summe: $\perp_{\text{sum}}(a,b) = a + b - a \cdot b$

Łukasiewicz: $\perp_{\text{Łuka}}(a,b) = \min\{1, a+b\}$

Drastische Summe: $\perp_{-1}(a,b) = \begin{cases} a, & \text{if } b = 0, \\ b, & \text{if } a = 0, \\ 1, & \text{otherwise.} \end{cases}$



Zusammenspiel der Fuzzy-Operatoren

- Es gilt $\forall a,b \in [0,1]: \quad \top_{-1}(a,b) \leq \top_{\text{Łuka}}(a,b) \leq \top_{\text{prod}}(a,b) \leq \top_{\min}(a,b)$.
  Auch alle anderen denkbaren t-Normen liegen zwischen $\top_{-1}$ und $\top_{\min}$.
- Es gilt $\forall a,b \in [0,1]: \quad \perp_{\max}(a,b) \leq \perp_{\text{sum}}(a,b) \leq \perp_{\text{Łuka}}(a,b) \leq \perp_{-1}(a,b)$.
  Auch alle anderen denkbaren t-Konormen liegen zwischen $\perp_{\max}$ und $\perp_{-1}$.
- Beachte: Es gilt i.a. *weder* $\top(a, \sim a) = 0$ *noch* $\perp(a, \sim a) = 1$.
- Ein Operatorensatz $(\sim, \top, \perp)$ bestehend aus einer Fuzzy-Negation $\sim$, einer t-Norm $\top$ und einer t-Konorm $\perp$ heißt **duales Tripel**, wenn mit diesen Operatoren die Verallgemeinerungen der DeMorganschen Gesetze gelten, d.h.

$$\forall a,b \in [0,1]: \quad \sim\top(a,b) = \perp(\sim a, \sim b)$$
$$\forall a,b \in [0,1]: \quad \sim\perp(a,b) = \top(\sim a, \sim b)$$

- Der am häufigsten benutzte Operatorsatz ist das duale Tripel $(\sim, \top_{\min}, \perp_{\max})$ mit der Standardnegation $\sim a \equiv 1 - a$.

Fuzzy-Mengenlehre

- Die klassische Mengenlehre basiert auf dem Begriff „*ist Element von*“ ($\in$).
  Alternativ kann man die Zugehörigkeit zu einer Menge mit einer *Indikatorfunktion* beschreiben:
  Sei X eine Menge. Dann heißt

$$I_M : X \to \{0,1\}, \qquad I_M(x) = \begin{cases} 1, & \text{falls } x \in X, \\ 0, & \text{sonst,} \end{cases}$$

  Indikatorfunktion der Menge M bzgl. der Grundmenge X.
- In der Fuzzy-Mengenlehre wird die Indikatorfunktion durch eine *Zugehörigkeitsfunktion* ersetzt:
  Sei X eine (klassische/scharfe) Menge. Dann heißt

$$\mu_M : X \to [0,1], \qquad \mu_M(x) \mathrel{\widehat{=}} \text{Zugehörigkeitsgrad von } x \text{ zu } M,$$

  Zugehörigkeitsfunktion (membership function) der **Fuzzy-Menge** M bzgl. der *Grundmenge* X.

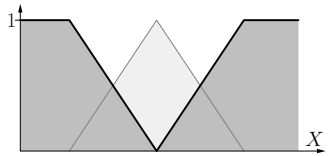
  Meist wird die Fuzzy-Menge mit ihrer Zugehörigkeitsfunktion identifiziert.

Fuzzy-Mengenlehre: Operationen

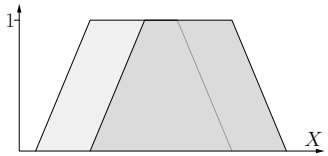
- Wie beim Übergang von der klassischen Logik zur Fuzzy-Logik ist beim Übergang von der klassischen Mengenlehre zur Fuzzy-Mengenlehre eine Erweiterung der Operationen nötig.
- **Grundprinzip dieser Erweiterung**:
  Greife auf die logische Definition der Operationen zurück.
  $\Rightarrow$ elementweise Anwendung der logischen Operatoren
- Seien A und B (Fuzzy-)Mengen über der Grundmenge X.

Komplement	klassisch	$\overline{A} = \{x \in X \mid x \notin A\}$
	fuzzy	$\forall x \in X: \quad \mu_{\overline{A}}(x) = \sim\mu_A(x)$
Schnitt	klassisch	$A \cap B = \{x \in X \mid x \in A \wedge x \in B\}$
	fuzzy	$\forall x \in X: \quad \mu_{A\cap B}(x) = \top(\mu_A(x), \mu_B(x))$
Vereinigung	klassisch	$A \cup B = \{x \in X \mid x \in A \vee x \in B\}$
	fuzzy	$\forall x \in X: \quad \mu_{A\cup B}(x) = \perp(\mu_A(x), \mu_B(x))$

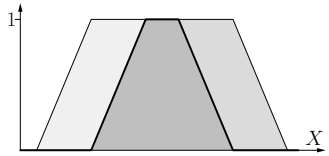
Fuzzy-Mengenoperationen: Beispiele



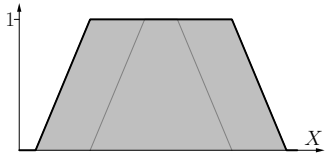
Fuzzy-Komplement



zwei Fuzzy-Mengen



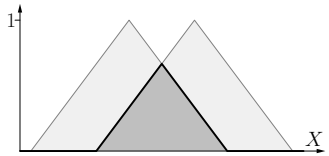
Fuzzy-Schnitt



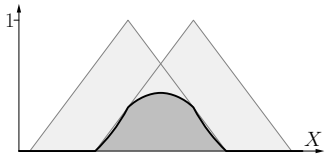
Fuzzy-Vereinigung

- Der links gezeigte Fuzzy-Schnitt und die rechts gezeigte Fuzzy-Vereinigung sind unabhängig von der gewählten t-Norm bzw. t-Konorm.

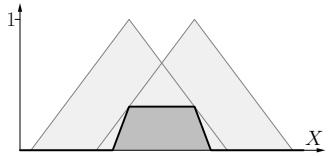
Fuzzy-Schnitt: Beispiele



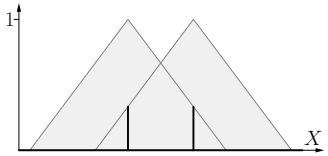
t-Norm $\top_{\text{min}}$



t-Norm $\top_{\text{prod}}$



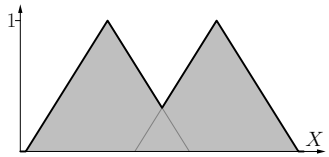
t-Norm $\top_{\text{Luka}}$



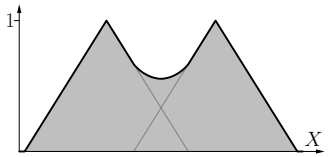
t-Norm $\top_{-1}$

- Man beachte, daß alle Fuzzy-Schnitte zwischen dem oben links und dem unten rechts gezeigten liegen.

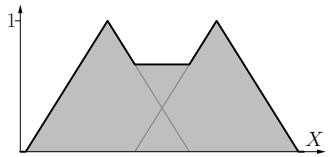
Fuzzy-Vereinigung: Beispiele



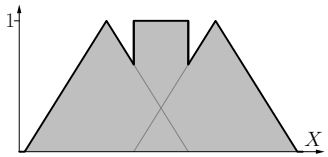
t-Konorm $\perp_{\text{max}}$



t-Konorm $\perp_{\text{sum}}$



t-Konorm $\perp_{\text{Luka}}$



t-Konorm $\perp_{-1}$

- Man beachte, daß alle Fuzzy-Vereinigungen zwischen der oben links und der unten rechts gezeigten liegen.

Fuzzy-Partitionen und Linguistische Variablen

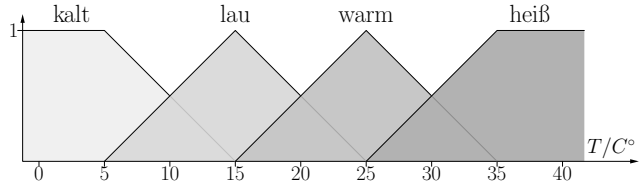
- Um einen Wertebereich durch sprachliche (linguistische) Ausdrücke beschreiben zu können, wird er mit Hilfe von Fuzzy-Mengen fuzzy-partitioniert.

  Jeder Fuzzy-Menge der Partitionierung ist ein linguistischer Term zugeordnet.

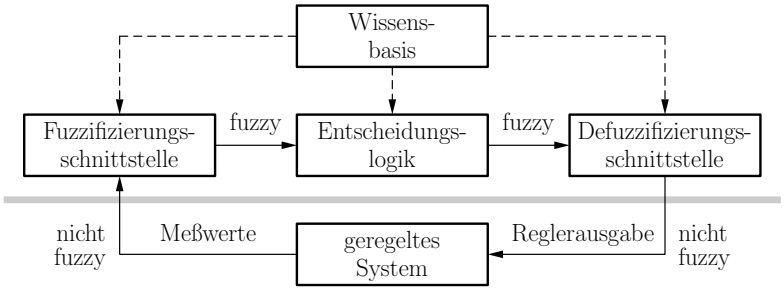
- Übliche Bedingung: An jedem Punkt müssen sich die Zugehörigkeitsgrade aller Fuzzy-Mengen zu 1 addieren *(partition of unity)*.

Beispiel: Fuzzy-Partitionierung für Temperaturen

Wir definieren eine linguistische Variable mit den Werten *kalt*, *lau*, *warm* und *heiß*.

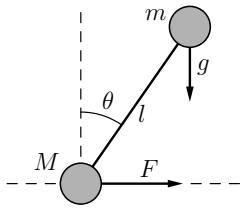


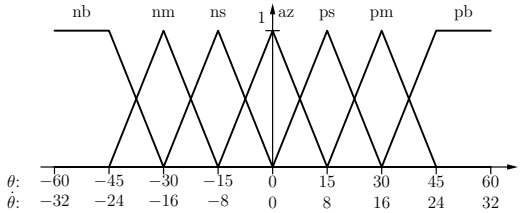
Architektur eines Fuzzy-Reglers



- Die Wissensbasis enthält die Fuzzy-Regeln für die Steuerung und die Fuzzy-Partitionen der Wertebereiche der Variablen.
- Eine Fuzzy-Regel lautet: **if** X_1 **is** $A_{i_1}^{(1)}$ **and** ... **and** X_n **is** $A_{i_n}^{(n)}$ **then** Y **is** B.
  $X_1, \ldots, X_n$ sind die Meßgrößen und Y ist die Stellgröße.
  $A_{i_k}^{(k)}$ und B sind linguistische Terme, denen Fuzzy-Mengen zugeordnet sind.

Beispiel-Fuzzy-Regler: Stabbalance



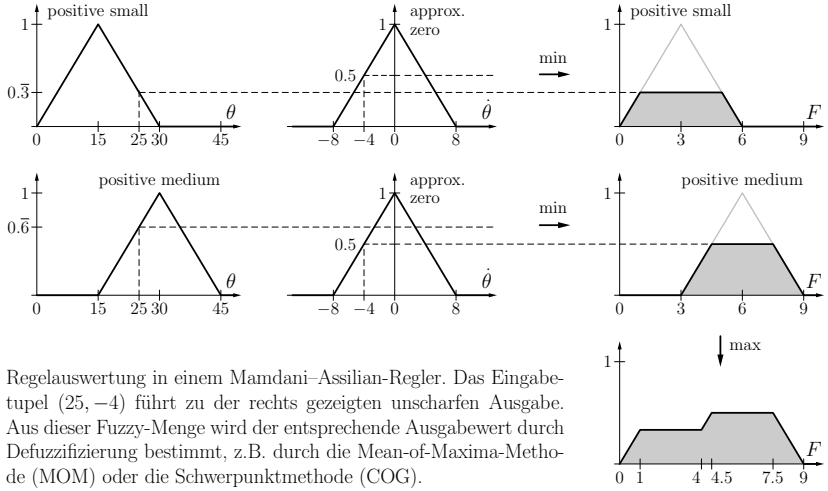


Abkürzungen

pb – positive big
pm – positive medium
ps – positive small
az – approximately zero
ns – negative small
nm – negative medium
nb – negative big

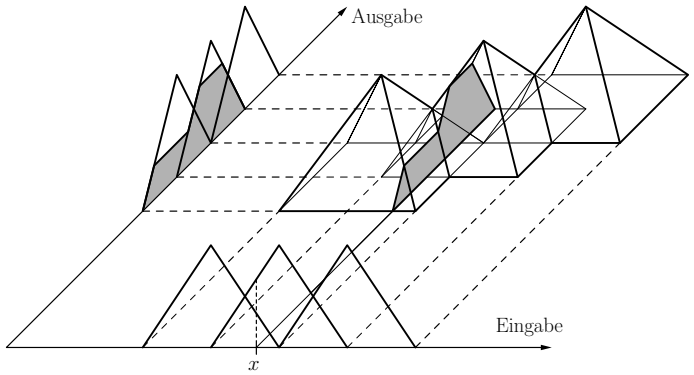
$\dot\theta \backslash \theta$	nb	nm	ns	az	ps	pm	pb
pb			**ps**	**pb**			
pm				**pm**			
ps	**nm**		**az**	**ps**			
az	**nb**	**nm**	**ns**	**az**	**ps**	**pm**	**pb**
ns				**ns**	**az**		**pm**
nm				**nm**			
nb				**nb**	**ns**		

Fuzzy-Regelung nach Mamdani–Assilian



Regelauswertung in einem Mamdani–Assilian-Regler. Das Eingabetupel $(25, -4)$ führt zu der rechts gezeigten unscharfen Ausgabe. Aus dieser Fuzzy-Menge wird der entsprechende Ausgabewert durch Defuzzifizierung bestimmt, z.B. durch die Mean-of-Maxima-Methode (MOM) oder die Schwerpunktmethode (COG).

Fuzzy-Regelung nach Mamdani–Assilian



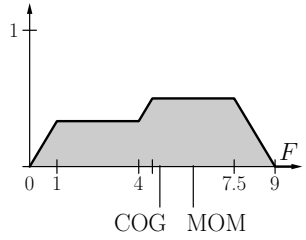
Ein Fuzzy-Regelsystem einer Meß- und einer Stellgröße und drei Fuzzy-Regeln.
Jede Pyramide wird durch eine Fuzzy-Regel spezifiziert.
Der Eingabewert x führt zu der grau gezeichneten unscharfen Ausgabe.

Defuzzifizierung

Die Auswertung der Fuzzy-Regeln liefert eine **Ausgabe-Fuzzy-Menge**.

Die Ausgabe-Fuzzy-Menge muß in einen **scharfen Stellwert** umgewandelt werden.

Dieser Vorgang heißt **Defuzzifizierung**.



Die wichtigstens Defuzzifizierungsmethoden sind:

- **Schwerpunktmethode** (Center of Gravity, COG)
  Der Schwerpunkt der Fläche unter der Ausgabe-Fuzzy-Menge.
- **Flächenmittelpunktmethode** (Center of Area, COA)
  Der Punkt, der die Fläche unter der Ausgabe-Fuzzy-Menge in gleich große Teile teilt.
- **Maxima-Mittelwert-Methode** (Mean of Maxima, MOM)
  Das arithmethische Mittel der Stellen mit maximalem Zugehörigkeitsgrad.

Erzeugen/Optimieren von Fuzzy-Reglern mit GA

- Bei einem Mamdani-Assilian-Regler kann optimiert werden:
  - Die **Regelbasis** (welche Regeln, welche Ausgaben).
  - Die **Fuzzy-Mengen/Fuzzy-Partitionen** (Form, Lage, Ausdehnung, ggf. Anzahl der Fuzzy-Mengen).
  - Die **t-Norm** bzw. **t-Konorm** für die Regelauswertung (selten).
  - Parameter der **Defuzzifizierungsmethode** (falls vorhanden; selten).
  - Welche **Eingangsgrößen** in den Regeln verwendet werden. (Merkmalsauswahl, engl. *feature selection*)
- Wir betrachten nur die Optimierung der Regelbasis und der Fuzzy-Mengen bei fester Wahl der Eingabe-/Meßgrößen.
- Die Regelauswertung geschieht (wie gezeigt) über Minimum und Maximum, die Defuzzifizierung über die Schwerpunktmethode.

Erzeugen/Optimieren von Fuzzy-Reglern mit GA

Mögliche Vorgehensweisen:

- *Die Regelbasis und die Fuzzy-Partitionen werden gleichzeitig optimiert.*

  Nachteil: Sehr große Zahl von Parametern muß gleichzeitig optimiert werden.
- *Erst werden die Fuzzy-Partitionen bei vorgegebener Regelbasis optimiert, dann wird die Regelbasis mit den besten Fuzzy-Partitionen optimiert.*

  Nachteil: Zum Aufstellen der Regelbasis muß Expertenwissen verfügbar sein. (Wahl einer zufälligen Regelbasis ist wenig erfolgversprechend.)
- *Erst wird die Regelbasis für vorgegebene Fuzzy-Mengen optimiert, dann werden die Fuzzy-Partitionen mit der besten Regelbasis optimiert.*

  Die Fuzzy-Mengen können z.B. äquidistant (gleichmäßig) verteilt werden. In diesem Fall muß ein Anwender lediglich die Zahl der Fuzzy-Mengen je Meßgröße und für die Stellgröße vorgeben.

  → Hier betrachten wir nur diese Möglichkeit.

Erzeugen/Optimieren von Fuzzy-Reglern: Fitneßfunktion

- Ein guter Regler sollte verschiedene Kriterien erfüllen:
  - Aus jeder möglichen Situation sollte der Sollzustand erreicht werden.
  - Der Sollzustand sollte möglichst schnell erreicht werden.
  - Der Sollzustand sollte mit geringem (Energie-)Aufwand erreicht werden.
- Der Regler wird mehrfach testweise auf das zu regelnde System angewandt.

  (hier: Simulation des Stabbalance-Problems in einem Rechner, mehrere, zufällig gewählte Anfangssituationen)

  Je nach Regelerfolg/Regelgüte erhält der Regler Punkte (Anzahl Situationen, Dauer erfolgreicher Regelung, Energieaufwand).
- **Beachte:**
  In dieser Anwendung ist die Bewertung der Individuen die mit Abstand aufwendigste Operation. Jedes Individuum muß über eine gewisse Mindestzahl von Zeitschritten zur Regelung eingesetzt werden.

Bewertung des Regelerfolgs

- Weicht der Istwert zu stark vom Sollwert ab, wird abgebrochen (Fehlschlag). (z.B. Stabbalance-Problem: der Istwert muß im Intervall $[-90°, 90°]$ bleiben.)
- Nach einer bestimmten Dauer sollte der Istwert in der Nähe des Sollwertes liegen und dort verbleiben (Toleranzbereich). Ist dies nicht der Fall, wird ebenfalls abgebrochen (Fehlschlag).
- Der Toleranzbereich wird im Laufe der Generationen verringert (langsames Hinführen auf das gewünschte Ziel).
  - In den ersten Generationen ist es hinreichend, wenn der zu balancierende Stab nicht umfällt.
  - Später muß der Stab in einem immer engeren Winkelbereich um die senkrechte Lage gehalten werden.
- Die Beträge der Stellwerte werden aufsummiert und als Strafterm verwendet.
  (In der Gleichgewichtslage hat einen schnelles Umschalten zwischen großen Kräften gleiche Wirkung wie eine Regelung mit geringen Kräften. Hohe Kräfte sollen vermieden werden.)

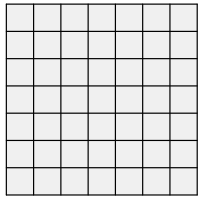
Erzeugen/Optimieren der Regelbasis: Kodierung

- Es werden nur vollständige Regelbasen betrachtet (zu jeder Kombination von Eingabe-Fuzzy-Mengen gibt es eine Regel).
- Für jede Kombination von Eingabe-Fuzzy-Mengen muß lediglich der linguistische Term der Stellgröße festgelegt werden (i.w. Ausfüllen einer Tabelle).

Beispiel: Regelbasis für Stabbalance-Regler:

$\dot{\theta}\backslash\theta$	nb	nm	ns	az	ps	pm	pb
pb	**az**	**ps**	**ps**	**pb**	**pb**	**pb**	**pb**
pm	**ns**	**az**	**ps**	**pm**	**pm**	**pb**	**pb**
ps	**nm**	**ns**	**az**	**ps**	**pm**	**pm**	**pb**
az	**nb**	**nm**	**ns**	**az**	**ps**	**pm**	**pb**
ns	**nb**	**nm**	**nm**	**ns**	**az**	**ps**	**pm**
nm	**nb**	**nb**	**nm**	**nm**	**ns**	**az**	**ps**
nb	**nb**	**nb**	**nb**	**nb**	**ns**	**ns**	**az**

schematisch



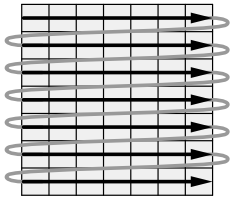
Kodierung der Regelbasis

Darstellung der Regelbasis als Chromosom

- **Linearisierung** (Umwandlung in Vektor)
  Die Tabelle wird in einer willkürlichen, aber festen Reihenfolge durchlaufen und die Tabelleneinträge werden in einem Vektor aufgelistet.

  Beispiel: zeilenweise Auflistung

  Problem: Nachbarschaftsbeziehungen zwischen den Zeilen gehen verloren
  (benachbarte Einträge sollten ähnliche linguistische Terme enthalten; dies sollte z.B. beim Crossover berücksichtigt werden)



- **Tabelle** (direkte Verwendung des Schemas)
  Es werden zwei- oder mehrdimensionale Chromosomen erzeugt.
  (In diesem Fall werden spezielle genetische Operatoren benötigt.)

Genetische Operatoren für die Regelbasis

- **Mutation:** (analog zu Standardmutation)
  - Eine Regel (ein Tabelleneintrag) wird zufällig gewählt.
  - Der linguistische Term der Ausgabe wird zufällig verändert.
  - Es können ggf. mehrere Regeln/Tabellenfelder gleichzeitig geändert werden.
- Es ist ggf. günstig, die Mutation einer Regelbasis so einzuschränken, daß ein Tabelleneintrag nur auf linguistische Terme geändert werden kann, die dem vorhandenen Eintrag benachbart oder hinreichend nahe sind.

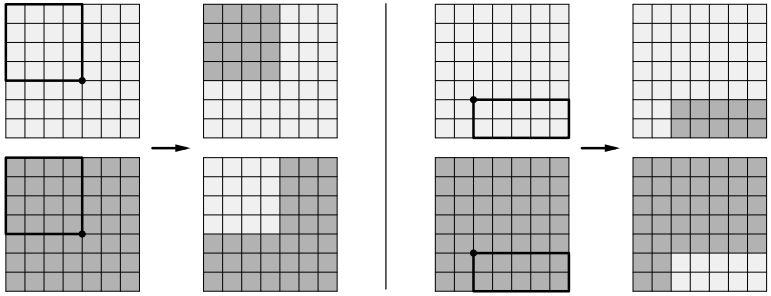
  Beispiele: „positive small“ kann nur auf „approximately zero“ oder „positive medium“ geändert werden.

  „negative big“ kann nur auf „negative medium“ oder ggf. „negative small“ geändert werden.

  (Dies verhindert ein zu schnelles „Zerfließen“ gesammelter Information; die Regelbasen werden nur „vorsichtig“ verändert.)

Genetische Operatoren für die Regelbasis

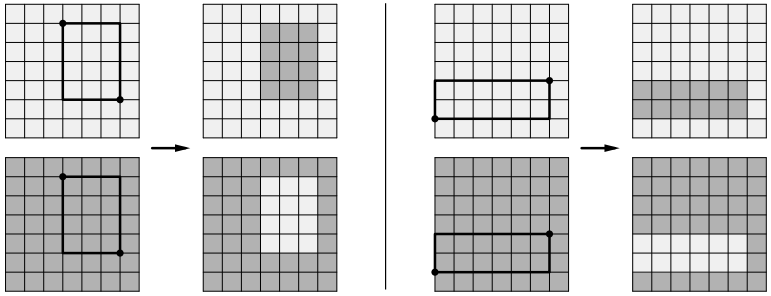
- **Crossover:** (Ein-Punkt-Crossover)
  Wähle zufällig einen inneren Gitterpunkt der Tabelle und eine Ecke.
  Tausche die so definierte Teiltabellen zwischen zwei Eltern aus.



- **Beachte:** Um bevorzugt benachbarte Regeln zusammen zu vererben, *sollte* das Crossover ortsabhängige Verzerrung zeigen!

Genetische Operatoren für die Regelbasis

- **Crossover:** (Zwei-Punkt-Crossover)
  Wähle zufällig zwei Gitterpunkte der Tabelle (auch Randpunkte).
  Tausche die so definierte Teiltabellen zwischen zwei Eltern aus.



- Zwei-Punkt-Crossover ist besser geeignet als Ein-Punkt-Crossover, da Teillösungen flexibler ausgetauscht werden können.

Optimierung der Fuzzy-Mengen

- **Gegeben:** optimierte Regelbasis mit fest gewählten und nicht veränderten äquidistanten (gleichmäßig verteilten) Fuzzy-Mengen.
- **Gesucht:** weitere Verbesserung des Reglerverhaltens durch Anpassen der Fuzzy-Mengen bei fester Regelbasis („Fine-Tuning“)
- **Kodierung der Fuzzy-Mengen:** (erste Möglichkeit)
  - Wähle die Form der Fuzzy-Mengen.
    (z.B. Dreieck, Trapez, Gaußglocke, Parabel, Spline etc.)
  - Liste die definierenden Parameter der Fuzzy-Mengen auf.
    (z.B. Dreieck: linker Rand, Mitte, rechter Rand)

Beispiel Stabbalance-Regler mit dreiecksförmigen Fuzzy-Mengen (Ausschnitt):

...	nm			ns			az			ps			...
...	−45	−30	−15	−30	−15	0	−15	0	15	0	15	30	...

Optimierung der Fuzzy-Mengen

Nachteile dieser Kodierung:

- Kodierung ist sehr „starr“ bzgl. der Form der Fuzzy-Mengen:
  Es wird z.B. vorher festgelegt, ob Dreiecke oder Trapeze verwendet werden.
- Genetische Operatoren können die Ordnung der Parameter zerstören
  (es muß bei Dreiecken z.B. gelten: linker Rand $\leq$ Mitte $\leq$ rechter Rand).
- Mögliche „Überholvorgänge“ zwischen Fuzzy-Mengen:
  Durch Mutation und Crossover kann die sinnvolle Reihenfolge der Fuzzy-Mengen zerstört werden (es sollte z.B. gelten: ns liegt rechts von ps).
- Die Bedingung „Summe der Zugehörigkeitsgrade = 1“ wird ggf. verletzt
  (kann durch einmalige Darstellung identischer Parameter behandelt werden).

...	−45	−15	15	−15	15	30	15	30	45	30	45	60	...
...	−45	−30	−20	−30	−20	−10	−20	−10	0	−10	10	30	...

Kodierung der Fuzzy-Partitionen

An einer Reihe von gleichmäßig über den Wertebereich verteilten Stützstellen werden die Zugehörigkeitsgrade der verschiedenen Fuzzy-Mengen angegeben.

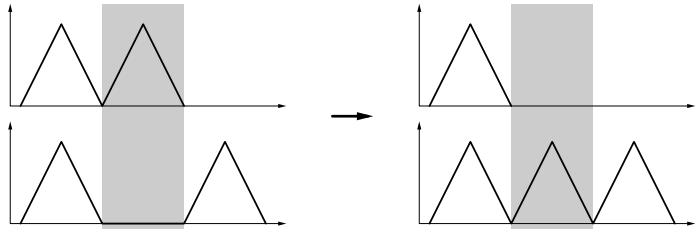
$$\underbrace{\overbrace{\begin{pmatrix} \mu_1(x_1) \\ \vdots \\ \mu_n(x_1) \end{pmatrix}}^{\text{Gen 1}} \cdots \overbrace{\begin{pmatrix} \mu_1(x_i) \\ \vdots \\ \mu_n(x_i) \end{pmatrix}}^{\text{Gen i}} \cdots \overbrace{\begin{pmatrix} \mu_1(x_m) \\ \vdots \\ \mu_n(x_m) \end{pmatrix}}^{\text{Gen m}}}_{\text{Kodierung mit } m \times n \text{ Zahlen aus } [0,1]}$$

pb	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	1	1	1
pm	0	0	0	0	0	0	0	0	0	0	0	0.5	1	0.5	0	0	0
ps	0	0	0	0	0	0	0	0	0	0.5	1	0.5	0	0	0	0	0
az	0	0	0	0	0	0	0	0.5	1	0.5	0	0	0	0	0	0	0
ns	0	0	0	0	0	0.5	1	0.5	0	0	0	0	0	0	0	0	0
nm	0	0	0	0.5	1	0.5	0	0	0	0	0	0	0	0	0	0	0
nb	1	1	1	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0
	-60		-45		-30		-15		0		15		30		45		60

Genetische Operatoren

- **Mutation:** analog zur Standardmutation
  Ein zufällig ausgewählter Eintrag wird zufällig geändert.

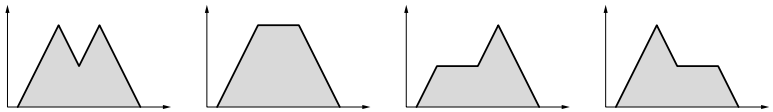
  Es ist sinnvoll, die Größe der Änderung zu begrenzen,
  z.B. durch Festlegen eines Intervalls oder durch eine Normalverteilung.
- **Crossover:** Einfaches Ein-Punkt- oder Zwei-Punkt-Crossover.
- Beachte: Durch Crossover können Fuzzy-Mengen ausgelöscht werden.



Reparatur der Fuzzy-Mengen

- Durch Mutation/Crossover kann der Fall eintreten, daß die Zugehörigkeitsfunktionen der Fuzzy-Mengen nicht mehr *unimodal* sind (d.h., die Zugehörigkeitsfunktion hat nur ein lokales Maximum.)

  Multimodale Fuzzy-Mengen sind viel schwerer zu interpretieren als unimodale.
- Daher werden die Fuzzy-Mengen ggf. repariert (unimodal gemacht).



bimodale Fuzzy-Menge | Verbinden der Maxima | Abschneiden des linken Maximums | Abschneiden des rechten Maximums

- Ggf. sind auch Fuzzy-Mengen zu verbreitern oder abzuschneiden, damit der gesamte Wertebereich mit Fuzzy-mengen überdeckt ist, aber nicht zu viele Fuzzy-Mengen den gleichen Bereich abdecken.

Erzeugen/Optimieren von Fuzzy-Reglern mit GA

- Erzeugen/Optimieren in zwei Schritten:
  - Optimieren der Regelbasis bei festen Fuzzy-Partitionen.
  - Optimieren der Fuzzy-Partitionen mit erzeugter Regelbasis.
- In Experimenten gelang es, mit einem genetischen Algorithmus funktionsfähige Fuzzy-Regler für das Stabbalance-Problem zu erzeugen.
- Diese Art der Reglererzeugung ist zwar sehr aufwendig (Rechenzeit), hat aber den Vorteil, daß man kaum Hintergrundwissen benötigt.
- Weitere Anforderungen, die man in der Fitneßfunktion berücksichtigen kann:
  - **Kompaktheit:** geringe Anzahl von Regeln und Fuzzy-Mengen
  - **Vollständigkeit:** Abdeckung relevanter Bereiche im Eingaberaum
  - **Konsistenz:** keine Regeln mit sehr ähnlichem Antezedens und verschiedenem Konsequens
  - **Interpretierbarkeit:** Beschränkte Überlappung von Fuzzy-Mengen