Stundenplanerstellung – Genetische Algorithmen

Wolfgang Barthel
Hannoversche Strasse 137a
37077 Göttingen
Tel.:(0551) 300 112
e-mail: barthel@math.uni-goettingen.de

2. Dezember 1999

Inhaltsverzeichnis

1 Einführung

Kein Mensch oder Gegenstand kann zur gleichen Zeit an zwei räumlich getrennten Orten sein und jeder hat nur endlich viel Zeit zur Verfügung. Um den Ansprüchen bei der Verteilung von Zeit gerecht zu werden ist es notwendig, die Zeit einzuteilen, d. h. einen Plan zu erstellen. Dieser Plan muss zum einen gewisse Randbedingungen erfüllen und andererseits möglichst vielen der Wünsche gerecht werden. Dazu zunächst das

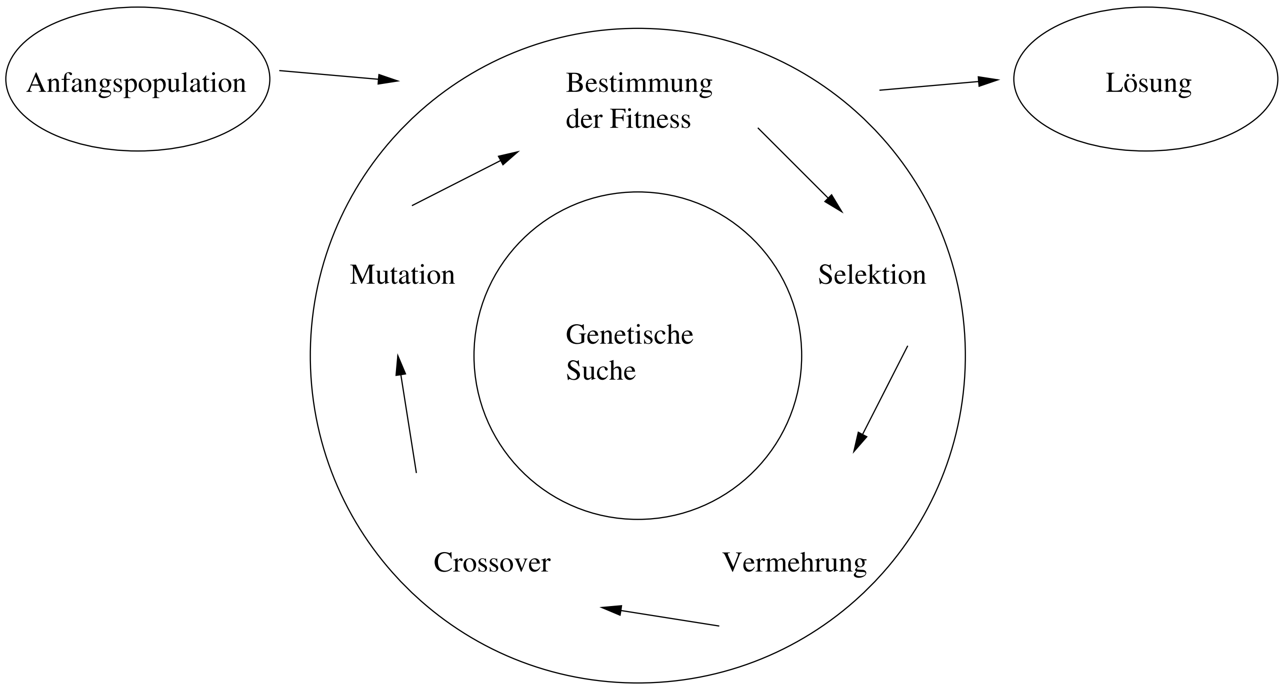
Beispiel 1. *Kurswahl in der Oberstufe im Gymnasium: Jeder Schüler muss eine bestimmte Auswahl von Kursen treffen, zwei fünfstündige und eine gewisse Anzahl dreistündige. Dann muss ein Stundenplan erstellt werden, an den als Mindestanforderung gestellt wird, das kein Schüler zwei Kurse zur selben Zeit besuchen muss und jeder Kurs die vorgeschriebene Länge hat.*

Die Lösungsmenge (Menge aller denkbaren Belegungen des Plans) ist bei solchen Problemen sehr groß, das Finden der idealen Lösung sehr zeitaufwendig (Beispiel ist schon NP-hard).

2 Genetische Algorithmen

Ein mögliches Lösungsverfahren versucht, den Evolutionsprozess in der Natur zu modellieren, anfänglich hauptsächlich genutzt, um diesen nachzuvollziehen. Eine Variante davon sind die genetischen Algorithmen. Jedes Element der Lösungsmenge entspricht einem möglichem *Individuum*, das ist eine Struktur (gewöhnlich Vektor aus $V_{\mathbb{F}_2}^N$). Jeder einzelne Eintrag des Vektors wird *Gen* genannt, dessen mögliche Zustände bezeichnet man als *Allel*. Eine *Population* bezeichnet eine Menge von ausgewählten *Individuen*.

Dem Algorithmus liegt folgendes Prinzip zu Grunde (nach [3]):



1. Initialisierung einer *Anfangspopulation*
   Es wird eine gewisse Anzahl n von möglichst verschiedenen Individuen erzeugt, entweder zufällig oder schon ausgehend von einer gewissen Vorstellung der endgültigen optimalen Lösung.

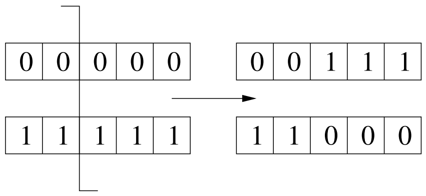
2. Evaluierung
   Mittels einer geeigneter *Fitness-Funktion F* werden die Individuen der aktuellen Population (*Eltern*) bewertet.

3. Selektion
   Auswahl von zur *Fortpflanzung* geeigneten Individuen in Abhängigkeit von der Fitnessfunktion (z.B. absolut nach Rangfolge oder mit einer Wahrscheinlichkeit relativ zur Fitness), die anderen werden verworfen.

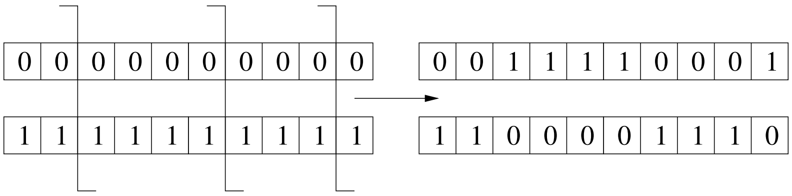
4. Vermehrung der Population (Erzeugung von *Kindern*) durch Anwendung genetischer Operatoren

   - Crossover
     Dieser Schritt ist der wichtigste Suchoperator, hier werden verschiedene Elemente verschiedener Lösungen kombiniert. Mit einer bestimmten Wahrscheinlichkeit P_{Cross} werden zwei Individuen der *Elterngeneration* miteinander *gekreuzt*. Dafür gibt es verschieden Möglichkeiten, deren Parameter zufällig ermittelt werden:

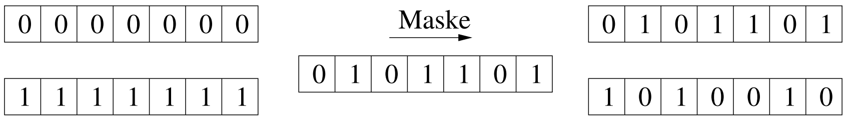
     – one-point-crossover



– n-point-crossover



– uniform-crossover



- Mutation
  Dieser genetische Operator dient dazu, durch Selektion in vorhergehenden Generationen verlorengegangene Allele wieder einzuführen. Jedes Gen eines jeden selektierten Individuums wird mit einer Wahrscheinlichkeit P_{Mut} invertiert.

5. Überprüfung des Abbruchbedingung
   Ist eine z.B. gewisse Qualität der Lösung erreicht (erneute Fortpflanzung bringt keine Verbesserung) oder eine bestimmte Anzahl von Iterationsschritten durchgeführt so wird der Algorithmus beendet und die optimalen Individuen ausgegeben.

Zur Verbesserung der Lösung kann(und sollte) der Algorithmus mehrfach abgearbeitet werden, je nach Güte der Zufallszahlen und Wahl der Parameter erhält man verschiedene Lösungen, unter denen man die optimale wählen kann.

3 Das Kurswahl-Problems

3.1 Formalisierung

Es bezeichne

K	die Menge der Kurse
S	die Menge der Schüler
U	die Menge der in Frage kommenden Unterrichtsstunden

sowie

$d : K \to \mathbb{N}$	die Dauer des Kurses $k \in K$
$w : S \to \mathcal{P}(K)$	die von Schüler $s \in S$ gewählten Kurse $k \in K$

Das zu lösende Problem hat dann folgende Gestalt:

Finde eine Abbildung $p : K \to \mathcal{P}(U)$ mit folgenden zwei Eigenschaften (Randbedingungen):

- $\forall k \in K : d(k) = \#\{u \in U : u \in p(k)\}$
  (d.h. jeder Kurs hat vorgeschriebene Stundenanzahl)
- $\forall s \in S : \sum\limits_{k \in w(s)} d(k) = \#\{u \in U : u \in p(w(s))\}$
  (d.h. kein Schüler muss zwei Kurse zur selben Zeit besuchen)

3.2 Beispielbelegung

Wir setzen

die Unterrichtstunden	$U = \{u_1, u_2, u_3, u_4, u_5\}$
die Kurse	$K = \{k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8\}$
mit entsprechender Dauer	$d((k_i)) = (3, 3, 2, 2, 2, 2, 2, 2)$
und die Schüler	$S = \{s_1, s_2, s_3, s_4, s_5, s_6\}$

Jeder Schüler wähle folgendermaßen je zwei Kurse:

$w(s_1) = \{k_1, k_5\}$	$w(s_2) = \{k_1, k_6\}$
$w(s_3) = \{k_2, k_3\}$	$w(s_4) = \{k_2, k_4\}$
$w(s_5) = \{k_1, k_8\}$	$w(s_6) = \{k_2, k_7\}$

3.3 Lösung

Zunächst ist notwendig, eine geeignete Repräsentation der Individuen zu erstellen. Dies geschieht mit der Stundenplanmatrix (x_{uk}), die definiert wird durch

$$x_{uk} = \begin{cases} 0, & \text{wenn } k \notin p^{-1}(u) \\ 1, & \text{wenn } k \in p^{-1}(u) \end{cases}$$

Für unser Beispiel wäre ein mögliches Individuum I_1 z.B.:

	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8
Stunde 1	1	1	0	0	1	0	0	0
Stunde 2	0	1	1	0	1	1	0	1
Stunde 3	1	1	0	0	0	0	0	0
Stunde 4	1	0	0	1	0	1	1	0
Stunde 5	0	0	1	1	0	0	1	1

Das Individuum erfüllt schon die erste Randbedingung, jedem Kurs sind so viele Stunden wie vorgeschrieben zugewiesen. Diese Eigenschaft sollen alle Individuen der Startpopulation haben. Auch die genetischen Operatoren werden so definiert, dass diese Eigenschaft erhalten bleibt.
Eine geeignete Fitnessfunktion F wäre hier die Summe der Anzahl der Überschneidungen von Kursen je Schüler. Zum Beispiel ist $F(I_1) = 3$.

3.4 Die genetischen Operatoren

Um neue Generation zu erzeugen, werden die genetischen Operatoren definiert:

- one-point-crossover

  Alle Spalten ab einer zufällig ausgewählten Spalte N werden zwischen den Eltern E_1 und E_2 getauscht:

$$\tilde{p_1}\left(k_i\right) = \begin{cases} p_1\left(k_i\right), & \text{für } i < N \\ p_2\left(k_i\right), & \text{für } i \geq N \end{cases}$$

  Zum Beispiel:

	Elter 1 (p_1)								Elter 2 (p_2)							
	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8
Stunde 1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Stunde 2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Stunde 3	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
Stunde 4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
Stunde 5	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	Fitness 12								Fitness 12							

  One-point-crossover mit N=3

	Kind 1 ($\tilde{p_1}$)								Kind 2 ($\tilde{p_2}$)							
	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8
Stunde 1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Stunde 2	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1
Stunde 3	1	1	1	0	0	0	0	0	1	1	0	1	1	1	1	1
Stunde 4	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0
Stunde 5	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0
	Fitness 2								Fitness 7							

- Mutation

  Damit die Randbedingung weiter erfüllt bleibt, werden hier zwei Elemente einer Spalte miteinander getauscht, z.B.

	Elter 1 (p_1)								Kind 1 ($\tilde{p_1}$)							
	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8
Stunde 1	1	1	0	0	[1]	0	0	0	1	1	0	0	[0]	0	0	0
Stunde 2	0	1	1	0	1	1	0	1	0	1	1	0	1	1	0	1
Stunde 3	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0
Stunde 4	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0
Stunde 5	0	0	1	1	[0]	0	1	1	0	0	1	1	[1]	0	1	1
	Fitness 3								Fitness 2							

3.5 Parameterwahl

Sehr wichtig ist die geeignete Wahl der Wahrscheinlichkeiten P_{Cross} und P_{Mut} von *Crossover* und *Mutation*, sowie der Anzahl der Startindividuen n und der zur Fortpflanzung ausgewählten Individuen r. Für die Anwendung der genetischen Operatoren sind 3 Varianten denkbar:

1. Crossover und Mutation unabhängig voneinander
```
begin
    dofor jedes der r ausgewählten Individuen
    begin
        pc := Zufallszahl aus [0, 1];
        if pc < P_cross then
        begin
            if schon Kandidat für crossover vorhanden
                then crossover durchführen
                else Individuum wird Kandidat für crossover
        end
        else
        begin
            dofor jede Spalte des Individuums
            begin
                pm := Zufallszahl aus [0, 1];
                if pm < P_Mut
                    then mutiere Spalte
            end
        end
    end
end
```

2. Entweder Crossover oder Mutation
```
begin
    dofor jedes der r ausgewählten Individuen
    begin
        pc := Zufallszahl aus [0, 1];
        if pc < P_cross then
        begin
            if schon Kandidat für crossover vorhanden
                then crossover durchführen
                else Individuum wird Kandidat für crossover
        end
        else
        begin
            N := zufällig gewählte Spalte des Individuums
            mutiere Spalte N
        end
    end
end
```

3. Nur Mutation
```
begin
    dofor jedes der r ausgewählten Individuen
    begin
        N := zufällig gewählte Spalte des Individuums
        mutiere Spalte N
    end
end
```

Welche der Variante am besten geeignet ist, lässt sich nur durch Probieren ermitteln. Für unser Beispiel erhält man als eine optimale Lösung:

	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8
Stunde 1	1	1	0	0	0	0	0	0
Stunde 2	1	1	0	0	0	0	0	0
Stunde 3	1	1	0	0	0	0	0	0
Stunde 4	0	0	1	1	1	1	1	1
Stunde 5	0	0	1	1	1	1	1	1

4 Zusammenfassung

Der beschriebene Algorithmus lässt sich auf andere Probleme übertragen, für die keine herkömmlichen Optimierungsalgorithmen bekannt sind, oder diese zu langsam sind, da bei genetischen Algorithmen keine Forderungen an die zu optimierende Funktion gestellt werden muss. Zwar wird man im Allgemeinen kein absolutes Optimum finden, dafür aber hinreichend schnell eine gute Näherung. Das Konzept kann noch erweitert werden, in dem man weitere „evolutionäre Elemente“ einfließen lässt, wie z.B. zeitweilige Separierung von Teilpopulationen etc.

Vorteile:

- breite Anwendungsmöglichkeiten
- gut geeignet für mehrdimensionale Optimierungsprobleme
- gefundene Lösung ist nahe am Optimum
- keine Einschränkungen an zu optimierende Funktion (z.B. Linearität, Stetigkeit etc.)
- kein Wissen über Struktur der Lösungsmenge notwendig
- parallelisierbar

Nachteile:

- Einhaltung von Randbedingungen kann nicht garantiert sondern nur durch Wahl der Fitnessfunktion bevorzugt werden
- heuristische Methode, keine Garantie das Optimum in endlicher (oder unendlicher) Zeit zu finden
- gefundene Lösung abhängig vom Zufallsgenerator und dessen Güte
- rechenintensiv
- abhängig von geeigneter Wahl der Parameter wie z.B. P_{Cross}
- zum Fine-Tuning der Lösung ungeeignet

Literatur

[1] Junginger, W.: Course Scheduling by Genetic Algorithm, Springer Verlag, 1992

[2] Biethan, J.; Nissen, V. : Evolutionary algorithms in management applications, Springer Verlag, 1995

[3] Pohlheim, Hatmut: GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with MATLAB, http://www.geatbx.com