  
  
Anwendungen der linearen Optimierung

Prof. Dr. M. Grütz

**Projekt:**  
Portierung einer Software um die Lauffähigkeit

auf aktuellen Betriebssystemen zu gewährleisten.

**Software:**

„Terminplanung – Genetische Algorithmen“

**Studenten:**Jan Asmuth (284986)  
Markus Bleß (285259)  
  
WIN6 SS 2013  
Konstanz, 23.05.2013

**Inhaltsverzeichnis**

[1. Einführung 3](#_Toc354493394)

[1.1. Aufgabenstellung 3](#_Toc354493395)

[1.2. Hard- und Software 3](#_Toc354493396)

[1.3. Genetische Algorithmen 3](#_Toc354493397)

[2. Programmablauf 4](#_Toc354493398)

[2.1. Modul Init 4](#_Toc354493399)

[2.2. Modul Strafpunkte 5](#_Toc354493400)

[2.3. Modul Checkopt 5](#_Toc354493401)

[2.4. Modul Paarung 5](#_Toc354493402)

[2.5. Modul Mutation 6](#_Toc354493403)

[3. Ein-/Ausgabe 8](#_Toc354493404)

[4. Ablaufplan des genetischen Algorithmus 9](#_Toc354493405)

[5. Verbesserungen zum Vorgänger 10](#_Toc354493406)

## Einführung

### Aufgabenstellung

Neuprogrammierung eines genetischen Algorithmus mit dessen Hilfe eine Terminplanungsaufgabe gelöst werden kann. Zur Verfügung stehen uns ein Artikel aus der Zeitschrift „Computerwoche 18 vom 6. Mai 1994, der in C++ erstellte Programmcode und eine Dokumentation von WIN8-Studenten der HTWG Konstanz von 1994. Da das Programm auf Windows 7 nicht lauffähig und 64-bit-kompatibel ist schreiben wir das komplette Programm in Java neu.

### Hard- und Software

Für die Neuprogrammierung verwenden wir die Entwicklungsumgebung Eclipse Juno (JavaSE-1.7). Zur Versionsverwaltung und zum einfachen Arbeiten am gemeinsamen Code haben wir das die Buildmanagement-System github verwendet.

### Genetische Algorithmen

Genetische Algorithmen ermöglichen eine Programmier- bzw. Problemlösetechnik, die sich am Evolutionsgeschehen der Natur mit ihren von Generation zu Generation weiterwirkenden Mutations- und Selektionsmechanismen orientiert. Es läuft also im Rechner eine Art Darwinismus ab.

Das Haupteinsatzgebiet sind algorithmisch nur schwer zu beschreibende Probleme, die konventionell nur mit sehr großem Rechenaufwand zu lösen sind.

„Wie leistungsfähig die Technik des genetischen Problemlösens bereits ist, illustriert ein Projekt der Universität Edinburgh. Während die Schotten bisher bis zu sieben Mannwochen benötigten, um einen Terminplan für die Hochschulprüfungen aufzustellen, löst ein genetisch arbeitendes Rechnerprogramm die gleiche Aufgabe nun mit besseren Resultaten, und zwar in einer Minute.“[[1]](#footnote-1)

## Programmablauf

### Modul Init

Das Modul Init hat die Aufgabe das Feld zum Zeitpunkt Null mit Daten vor zu belegen, d.h. es wird die Ur-Entwicklungsstufe der Lebewesen erzeugt. Diese Vorbelegung erfolgt mit Hilfe des gleichen Zufallsprinzips, wie die Restbelegung nach der Mutation.

### Modul Strafpunkte

Das Modul Strafpunkte bewertet die Lebewesen nach ihrer Fitness. Dabei werden in diesem Programm die Strafpunkte wie folgt bewertet:

Termine am selben Tag 3 Strafpunkte

Termine an darauffolgenden Tagen 1 Strafpunkt

Die Strafpunkte werden für jedes Lebewesen aufsummiert und in einem Array gespeichert.

### Modul Checkopt

Im Modul Checkopt wird geprüft, ob die Evolution, schon ein optimales Lebewesen hervorgebracht hat. Optimal ist in unserem Beispiel ein Lebewesen, das keinen Strafpunkt (s.o.) besitzt (d.h. Nur ein Termin am selben Tag und keine 2 Tage hintereinander verschiedene Termine).

### Modul Paarung

Das Modul Paarung setzt sich aus den beiden Funktionen „**Auswahl des Partners“** und   
**„0/1-Bitstring**“ erzeugen zusammen.

**Auswahl des Partners:**

Zur Weiterentwicklung der Lebewesen ist ein „**Vater“** (Bestes Lebewesen) und ein Partner („**Mutter“**) notwendig. Der Partner wird zufällig aus der Menge der Lebewesen ausgewählt. Hierbei muss darauf geachtet werden, dass es sich nicht um das beste Lebewesen (**Bestleb**, also der Vater selbst), das schlechteste Lebewesen (**Worstleb**) oder das zweitschlechteste Lebewesen (**Worst2leb**) handelt.

Hinweis: Die Anzahl der Lebewesen (MAXLEB) muss mindestens 4 betragen, damit eine Paarung mit einem neutralen Partner stattfinden kann.

”Bei der Paarung sollte vermieden werden, nur das jeweils beste Lebewesen mit dem zweitbesten zu paaren; es würden dann die Gene des ‘besten’ Lebewesen den Genpool überschwemmen und die Evolution käme ins Stocken. Besser ist es, das beste Lebewesen mit einem anderen zufällig ausgewählten zu paaren.”[[2]](#footnote-2)

**0/1-Bitstring:**

Zur Paarung wird ein Bitstring benötigt, dessen Länge der Anzahl der Gene (Termine bzw. Prüfungstagen) entspricht. In diesem Bitstring wird zufallsmäßig an jede Position '0’ oder ‘1’ geschrieben. Zur genauen Bedeutung dieses Bitstrings siehe Modul Mutation.

„Eine Eigenschaft der genetischen Algorithmen ist, dass der Zufall eine bedeutende Rolle spielt. Sie sind nicht unbedingt deterministisch und wenn man Pech hat, terminiert der Algorithmus erst nach unendlich langer Zeit. Glücklicherweise geht auch hier die Wahrscheinlichkeit für diesen Fall gegen Null, so dass in der Praxis genetische Algorithmen zwar terminieren, es jedoch beliebig lange dauern kann, bis ein geeignetes Lebewesen auftritt.“[[3]](#footnote-3)

### Modul Mutation

Bei dem Modul Mutation geht es um die Weiterentwicklung (Mutation) der Lebewesen. Dabei sterben die beiden schlechtesten Lebewesen bzw. werden durch zwei neue Kinder ersetzt.

Das beste Lebewesen jeder Versuchsreihe bzw. Generation wird zum Ausgangspunkt oder Erzeuger für die nachfolgenden Iterationen. Es wird also auf den ‘Erfahrungsschatz’ der vorhergehenden Versuche aufgebaut.

Die Weiterentwicklung läuft zusammen mit dem Tod der beiden ‘schlechtesten’. Die Lebewesen, die sterben sollen, werden einfach durch die beiden neugeborene Lebewesen ersetzt. So erspart man sich das Zwischenspeichern redundanter Daten.

**Beispiel mit 5 Lebewesen an 5 Tagen mit 5 Terminen:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | (Strafpunkte) |
| Worst2leb1: | 0 | 3 | 0 | 2 | 0 | 9 |
| Bestleb: | 2 | 0 | 1 | 1 | 1 | 5 |
|  | 1 | 0 | 1 | 0 | 3 | 6 |
| Worstleb1 | 0 | 4 | 1 | 0 | 0 | 12 |
| Partner2 | 1 | 0 | 3 | 0 | 1 | 6 |

1 die beiden schlechtesten Lebewesen sterben

2 der Partner („Mutter“) wurde zufällig ausgewählt

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| 0/1-Bitstring | 0 | 1 | 1 | 0 | 1 |  |
|  |  |  |  |  |  |  |
| **Nach der Mutation:** | | | | | |  |
| \* Kind 2 3 | 1 | ~ | ~ | 0 | ~ |  |
|  | 2 | 0 | 1 | 1 | 1 |  |
|  | 1 | 0 | 1 | 0 | 3 |  |
| \* Kind 1 4 | ~ | 0 | 1 | ~ | 1 |  |
|  | 1 | 0 | 3 | 0 | 1 |  |
|  |  |  |  |  |  |  |

3 Beim Kind 2 werden noch 4 Klausuren per Zufall auf die freien Tage verteilt

4 Beim Kind 1 werden noch 3 Klausuren per Zufall auf die freien Tage verteilt

Das **Kind 1** nimmt den Platz des ehemals schlechtesten Lebewesens (**Worstleb**) ein. Dabei werden vom **Vater** (Bestleb) diejenigen Positionen übernommen, an denen im 0/1-Bitstring eine ‘1’ steht.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Vater** | 2 | 0 | 1 | 1 | 1 |  |
|  |  |  |  |  |  |  |
| 0/1-Bitstring | 0 | 1 | 1 | 0 | 1 |  |
|  |  |  |  |  |  |  |
| **Kind 1** | ~ | 0 | 1 | ~ | 1 |  |

Analog wird mit dem **Kind 2** verfahren. Dieses nimmt den Platz des zweitschlechtesten Lebewesens (**Worst2leb**) ein. Es übernimmt diejenigen Positionen des ”**Partners**”, an denen im 0/1-Bitstring eine ‘0’ steht.

Abschließend werden die fehlenden Klausuren mit dem Zufallsgenerator auf die noch freien Tage verteilt. Es muss natürlich darauf geachtet werden, dass insgesamt genau die Anzahl der gewünschten Prüfungen erreicht wird - nicht mehr und nicht weniger!

Es wird nun wieder in das Modul Strafpunkte verzweigt und damit beginnt der Zyklus wieder von vorn.

Das mehrfache Durchlaufen des Zyklus liefert (zumindest in der Theorie) immer bessere Zwischenergebnisse und am Ende eine perfekte Lösung.

## Ein-/Ausgabe

Eingabe:

Die Eingangsgrößen Tage (Anzahl der Prüfungstage), Termine (Anzahl Prüfungen) und die Anzahl der Generationen kann der Benutzer variabel eingeben.



Ausgabe:

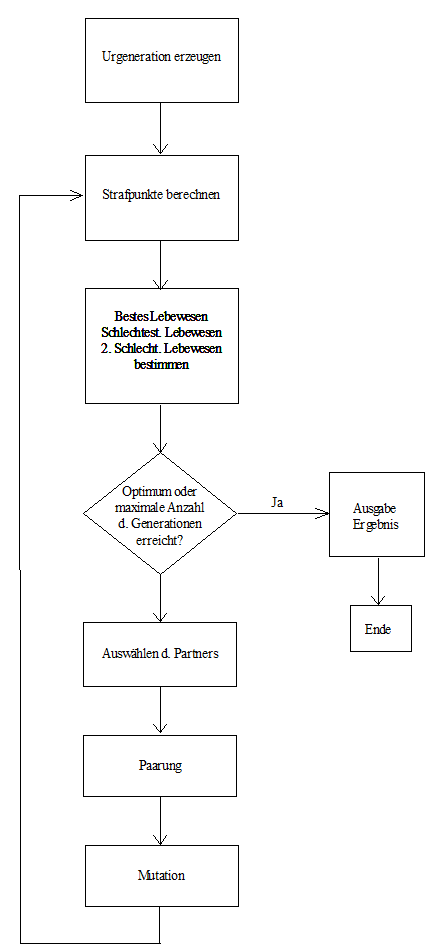
Als Ausgabe erhält man das Optimum oder das beste Lebewesen, mit den entsprechenden Terminen. Es gibt oft den Fall, dass es kein optimales Lebewesen geben kann.

**Bsp.**: Der Benutzer möchte, dass in 5 Tagen 6 Prüfungen stattfinden. Hier muss es an einem Tag 2 Prüfungen geben. Dies führt zu Strafpunkten und damit zu einem nicht optimalen Lebewesen.

In diesem Fall wird das Beste Lebewesen mit seiner Anzahl Strafpunkte ausgegeben.

Es wird daraus deutlich, dass es wichtig ist, die Anzahl der Generationen zu begrenzen, da es sonst eine unendliche Suche nach einem Optimum geben würde.

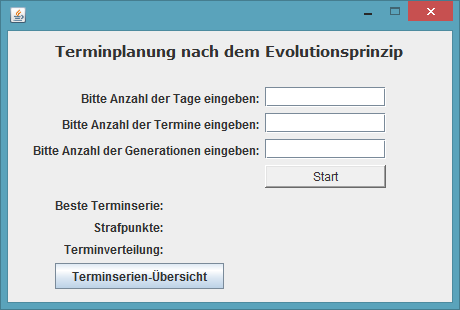
## Ablaufplan des genetischen Algorithmus



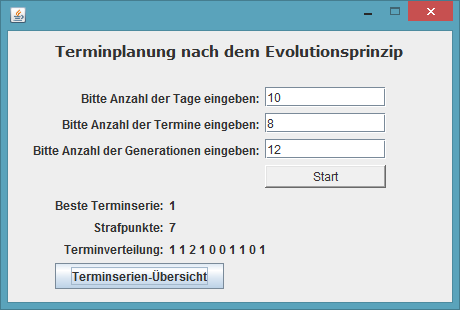
## Verbesserungen zum Vorgänger

Das Programm ist nun lauffähig unter Windows 7 (64-bit) und Windows 8, da es komplett in Java neu geschrieben wurde. Wir haben uns an den Prinzipien von Kapselung und objektorientiertem Programmieren gehalten. Das Programm wird nicht länger an der Konsole ausgeführt, sondern verfügt nun über eine benutzerfreundliche GUI. Es wird nicht nur noch die beste Terminserie angezeigt, sondern kann auf Wunsch auch alle Termine mit ihren Strafpunkten angezeigt werden.

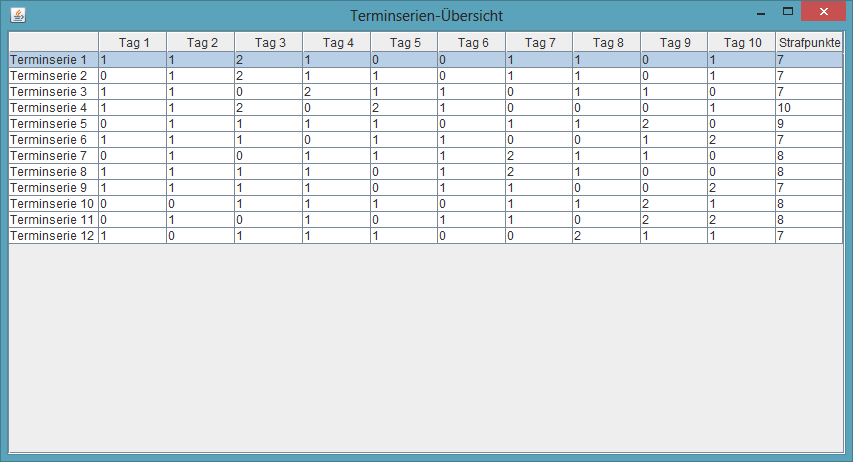
Wird die .jar-Datei gestartet wird der Benutzer gefragt über wie viele Tage die Termine verteilt sein sollen und wie viele Termine verteilt werden sollen. Durch die dritte Eingabe wird bestimmt wie oft Paarungen nach dem darwinistischen Prinzip stattfinden sollen. Eine höhere Zahl sorgt für einen insgesamt bessere Genpool und damit eine höhere Wahrscheinlichkeit eine optimalere Lösung zu finden.



Durch das Klicken auf den Start-Knopf wird der Algorithmus zur Simulation der Evolution zur Ermittlung der bestmöglichen Terminserie gestartet. Der Benutzer kann darauf auf den ersten Blick sehen, welche berechnete Terminserie die Beste ist, wie viele Strafpunkte sie hat und wie die Termine verteilt sind.



Klickt der Benutzer auf den Knopf "Tabelle" so wird eine Übersicht über den gesamten entstandenen Genpool nach der eingegebenen Anzahl von Generationen angezeigt. Die beste Terminserie wird markiert und in der rechten Spalte werden die zu jeder Terminserie gehörigen Strafpunkte angezeigt.



1. Computerwoche 18, 6.Mai 1994, S. 44 [↑](#footnote-ref-1)
2. c’t Heft 6/1989, S. 132 [↑](#footnote-ref-2)
3. c’t Heft 6/1989, S. 130 [↑](#footnote-ref-3)