Dokumentation der Chromosomendateien

(MEM und AKS Files)

GLEAM

Version 1.3

Wilfried Jakob
KIT, Campus Nord, Institut für Automation und angewandte Informatik (IAI)
PHermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany
email: wilfried.jakob@partner.kit.edu

Inhalt

Änderungen:

Änderungen gegenüber V1.0:
Einleitung

Änderungen gegenüber V1.1:

1. Einleitung
2. Ab der GLEAM-Version 2.2.2 (ab 28.6.2019) werden bei textuellen Chromosomenfiles nur noch diejenigen Genparameter geschrieben und gelesen, die im Genmodell spezifiziert sind. Damit können ältere textuelle Chromosomenfiles ab der Version 2.2.2 nicht mehr gelesen werden! Dies betrifft binäre Chromosomendateien nicht.
3. Verhalten bei gesetztem Compilerschalter `WITH_DYN_GENES`.

Änderungen gegenüber V1.2:
Redaktionelle Anpassungen an das Format der Dateidokumentationen

1 Einleitung

Chromosomenfiles gibt es als Binär- und als Textfiles, wobei letztere ursprünglich zum Export über Rechner- und/oder BS-Grenzen gedacht waren. Beide Dateiformate können interaktiv geladen werden, wobei das Fileformat beim Ladevorgang angegeben werden muss. Beim initialen Laden eines Chromosomenfiles durch Angabe in der EXP-Datei wird als Default davon ausgegangen, dass es sich um eine Binärdatei handelt. Andernfalls ist dies durch nachstehende Parametervorgabe in der TSK-Datei (siehe [1], Abschnitte 2.2 und 3.2) zu ändern:

```
Mit initialem textuellem Chr.File  = 1 bzw.
with initial textual chrom. file   = 1
```

Ein Chromosomenfile besteht aus einem oder mehreren Chromosomen. Die Dateien haben die Dateierweiterung MEM oder AKS, welche nur die Behandlung beim Laden durch GLEAM beeinflusst:

- MEM: Laden aller Chromosome der Datei in einen leeren Chromosomenspeicher. Wenn letzterer nicht leer ist, muss er zuvor per Dialog geleert werden.
- AKS: Laden aller Chromosome der Datei, wobei der Inhalt des Chromosomenspeichers erweitert wird.

Files beider Dateiformate können jeweils konkateniert werden. Man kann also z.B. eine textuelle MEM-Datei mit einer textuellen AKS-Datei zusammenfügen. Chromosomendateien sind immer im Zusammenhang mit dem zugehörigen Genmodell (MOD-File, siehe auch [2]) zu sehen. Der Versuch einer Interpretation eines Chromosoms im Kontext eines nicht dazu passenden Genmodells kann zu fehlerhaften und überraschenden Resultaten führen!

Wenn unsimulierte Chromosome geladen werden, erfolgt deren Simulation und Bewertung, so dass sie korrekt im Chromosomenspeicher abgelegt werden können und zur Initialisierung einer Startpopulation zur Verfügung stehen[1].

Besonderheiten ab Version 2.2.2:

Ab Version 2.2.2 gibt es für textuelle Chromosomenfiles ein neues Format, wobei der Unterschied in einer kompakteren Speicherung der Gene liegt. Daher enthält der dritte Abschnitt nur eine überarbeitete Version der Unterabschnitte 2.1.4 (Genelemente) und 2.2 (Beispiele). Der Rest der Beschreibung des Abschnitts 2 hat weiterhin Gültigkeit.

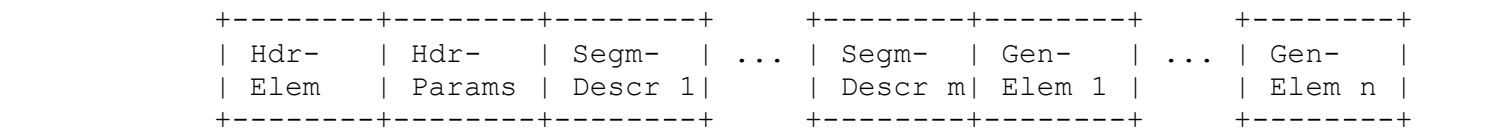
Außerdem können die Programme mit dem aktivierten Schalter `WITH_DYN_GENES` compiliert werden, wodurch die Beschränkungen der Größe eines Gens entfallen, also der maximalen Anzahl an Integer- und Realparametern, so wie sie in der Datei `chaindef.h` festgelegt ist. Es ist zu beachten, dass bei gesetztem `WITH_DYN_GENES` auch nur noch textuelle Chromosomendateien geladen oder geschrieben werden können.

2 Aufbau einer Chromosomen-Textdatei bis (Hy)GLEAM-Version 2.2.1

Die Textdatei enthält ein oder mehrere Chromosome hintereinander, wobei zwischen den Elementen eines Chromosoms und den Chromosomen selbst jeweils Zeilenwechsel stattfinden.

1: Das funktioniert bei einer Nutzung der externen Simulationsdienste (ESS) nur dann, wenn Einzelsimulationen möglich sind und sollte daher frühzeitig getestet werden. Z.B. in dem man das Statusfeld im Chromosom zurücksetzt, was sich bei textuellen Chromosomenfiles leicht machen lässt.

Ein Chromosom besteht aus einem *Header-Element*, einem *Header-Parameter-Element*, ein oder mehreren *Segmentdescriptoren* und einem oder mehreren *Genelementen*. Es hat folgenden Aufbau:



Jedes Element steht in einer (langen) Zeile.

2.1 Aufbau der Elemente

Die hier verwendeten KONSTANTEN und Variablen sind in der Datei `chaindef.h` im `sources`-Verzeichnis enthalten.

2.1.1 Header-Element

Durch mindestens ein Blank getrennte Einträge:

30200	Fixe Kennung (`activity`)
<fitness>	bei unsimulierten Chromosomen 0 (`fitness_note`)
<refs>	Anzahl der Referenzen auf dieses Chromosom, 0 (`ref_counter`)
<laenge>	Anzahle der Gene des Chromosoms (`chain_length`)
<guete>	bei unsimulierten Chromosomen 0 (`guete`)
<lfd-nr>	bei unsimulierten Chromosomen 0 (`lfd_nr`)
<status>	Statusbits des Chromosoms, 0 (`chain_status`)
<segm-anz>	Anzahl der Segmente des Chromosoms (`segment_anz`)
<ak-idx>	interne Verwaltung, kann 1 sein (`ak_index`)
12345	Terminator

Statusbits:

```
BASIS_AK                  1
TO_BE_SAVED               2
SIMULATED                 4
RANDOMLY_GENERATED        8
UNBEWERTETE_BASIS_AK  16384
UNMUTIERTE_URKETTE    32768
```

Anmerkung: Basis AKs spielen keine Rolle mehr.

2.1.2 Header-Parameter-Element

Dieses Element enthält die Simulationsergebnisse der letzten Simulation des Chromosoms und besteht aus 16 (`AK_ERG_WERTE_MAX`) Double-Werten. Bei unsimulierten Chromosomen sind das 16 Nullen.

2.1.3 Segmentdescriptoren

Segmentdescriptoren enthalten die Segmentstruktur in Form der Segmentlängen. Jeder Descriptor enthält 12 (`SEGM_PTR_ANZ`) ganzzahlige Werte. Für die im Kopf angegebene Segmentanzahl müssen sinnvolle Segmentlängen in den Segmentdescriptor-Elementen stehen. Sinnvoll heißt, dass sie jeweils größer als Null sind und dass ihre Summe der Kettenlänge entspricht. Wenn diese Vorgabe nicht eingehalten wird, gibt es beim Laden einen Fehler!

Zu beachten ist, dass der erste Wert des ersten Segmentdescriptor-Elements unbenutzt ist und durch eine Null dargestellt wird. Ein nicht vollständig genutztes Segmentdescriptor-Element wird mit Nullen aufgefüllt.

2.1.4 Genelemente

Ein Genelement besteht aus dem Gentyp (`activity`), den Genparametern und der Pointerkennung. Es enthält die Werte aller möglichen Genparameter unabhängig davon, ob das Genmodell diese vorsieht oder nicht. Unbenutzte Werte werden durch Nullen dargestellt. Somit besteht der Parameterteil eines Genelements immer aus 12 (`I_PAR_ANZ_MAX`) Integer-Werten und 8 (`R_PAR_ANZ_MAX`) Double-Werten.

Ein Genelement wird durch seine Pointerkennung terminiert. Diese beträgt 12345 für alle Genelemente außer dem letzten Genelement eine Chromosoms, welches mit 10000 terminiert wird.

2.2 Beispiele

Zuerst eine Datei mit zwei Chromosomen der math. Benchmarkfunktion "Foxholes", bei der es nur zwei Gentypen gibt, die jeweils nur einen Double-Parameter haben. Die Chromosomenlänge beträgt jeweils 2.

```
30200  0.000000000000e+00 0 2 0 0 0 2 0 12345              Chr1, Header
  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0                       Chr1, Headerparameter
  0 1 1 0  0 0 0 0  0 0 0 0                               Chr1, Segmentdescriptor
1  0 0 0 0  0 0 0 0  0 0 0 0  0.5  0 0 0  0 0 0 0  12345  Chr1, 1.Gen: Gentyp 1
2  0 0 0 0  0 0 0 0  0 0 0 0  22.1 0 0 0  0 0 0 0  10000  Chr1, 2.Gen: Gentyp 2
30200  0.0 0 2 0 0 0 1 0 12345                            Chr2, Header
  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0                       Chr2, Headerparameter
  0 2 0 0  0 0 0 0  0 0 0 0                               Chr2, Segmentdescriptor
2  0 0 0 0  0 0 0 0  0 0 0 0   63.5 0 0 0  0 0 0 0  12345  Chr2, 1.Gen: Gentyp 2
1  0 0 0 0  0 0 0 0  0 0 0 0  -22.1 0 0 0  0 0 0 0  10000  Chr2, 2.Gen: Gentyp 1
```

Das erste Chromosom hat zwei Segmente der Länge 1. Das zweite nur 1 Segment der Länge 2. Beide Chromosome haben keine Adresse im Chromosomenspeicher (<guete> und <lfd-nr> sind beide Null) und das Statusfeld ist nicht gesetzt. Letzteres würde beim Einlesen beim Programmstart (Angabe der Datei im Experimentfile als Wert des Eintrags `Chromosomenspeicher`) dazu führen, dass sie simuliert und bewertet werden, damit sie korrekt im Speicher einsortiert werden können. Die nachfolgende 2 gibt an, dass das Chromosom aus zwei Segmenten besteht. Die Gene des 2. Chromosoms sind in umgekehrter Reihenfolge zur Definition im Genmodell.

Reelle Zahlen, die Null sind, werden als 0 geschrieben. Ansonsten im e-Format mit 12 Nachkommastellen (aus Platzgründen in den Beispielen verkürzt). Beim Einlesen ist natürlich jedes zulässige Format für Double-Werte entsprechend den Konventionen von C zulässig.

Das zweite Beispiel bezieht sich auf das Robotergenmodel (siehe Genmodelldatei `mitsu_hy.mod` im Verzeichnis `InitFiles`), das Chromosomen dynamischer Länge beinhaltet. Dargestellt wird ein unsimuliertes und damit unbewertetes Chromosom der Länge 5. Die simulierte Variante ist dann im zweiten Beispiel des Abschnitts 3.2 enthalten.

```
30200  0.000000000000e+00 1 5 1 1 8 2 1  12345
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 2 3 0 0 0 0 0 0 0 0 0
  1   0 0 0 0 0 0 0 0 0 0 0 0  7.20301924e-01 -7.7605766e+01 0 0 0 0 0 0  12345
  8   0 0 0 0 0 0 0 0 0 0 0 0  1.18683428e+00 0 0 0 0 0 0 0 0  12345
  2   0 0 0 0 0 0 0 0 0 0 0 0  7.35953664e+00  6.4200671e+00 0 0 0 0 0 0  12345
 10   0 0 0 0 0 0 0 0 0 0 0 0  2.87482336e+00 0 0 0 0 0 0 0 0  12345
 13  19 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0    10000
```

Die Fitness ist Null, das Chromosom besteht aus fünf Genen und stand unter der Adresse 1/1 im Chromosomenspeicher. Im Status ist das Bit 8, `RANDOMLY_GENERATED` gesetzt. Das Chromosom hat zwei Segmente der Länge 2 und 3, wie aus dem Segmentdeskriptor hervorgeht. Da das Chromosom unsimuliert ist, sind alle Werte des Headerparameterelements Null. Die Gentypen 1 und 2

haben 2 reelle Genparameter und die 8 und 10 hingegen nur einen. Das Gen vom Typ 13 besteht nur aus einem ganzzahligen Parameter.

3 Aufbau der Chromosomen-Textdatei ab (Hy)GLEAM-Version 2.2.2

Die Unterschiede beider Versionen betreffen nur die Genelemente (Abschnitt 2.1.4) und das Beispiel (Abschnitt 2.2) . Alles andere zuvor Gesagte ist auch in Version 2.2.2 und folgende gültig.

3.1 Genelemente

Ein Genelement besteht aus dem Gentyp (`activity`), den Genparametern und der Pointerkennung. Es enthält die Werte der Genparameter, welche im Genmodell definiert wurden. Im Gegensatz zu früheren Versionen gibt es keine unbenutzten Werte mehr, welche durch Nullen dargestellt würden.

Ein Genelement wird durch seine Pointerkennung terminiert. Diese beträgt wie bisher auch 12345 für alle Genelemente außer dem letzten Genelement, welches mit 10000 terminiert wird.

3.2 Beispiele

Zuerst die Darstellung der gleichen zwei Chromosome des vorangegangenen Beispiels für die Version 2.2.1 im neuen Dateiformat. Wie zuvor handelt es sich um 2 Chromosomen der math. Benchmarkfunktion "Foxholes", bei der es nur 2 Gentypen gibt, die jeweils nur einen Double-Parameter haben. Die Chromosomenlänge beträgt jeweils 2.

```
30200  0.000000000000e+00 0 2 0 0 0 2 0 12345        Chr1, Header
  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0                 Chr1, Headerparameter
  0 1 1 0  0 0 0 0  0 0 0 0                          Chr1, Segmentdescriptor
1  5.000000000000e-01  12345                        Chr1, 1.Gen: Gentyp 1
2  2.210000000000e+01  10000                        Chr1, 2.Gen: Gentyp 2
30200  0.0 0 2 0 0 0 1 0 12345                       Chr2, Header
  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0                 Chr2, Headerparameter
  0 2 0 0  0 0 0 0  0 0 0 0                          Chr2, Segmentdescriptor
2  6.350000000000e+01  12345                        Chr2, 1.Gen: Gentyp 2
1  -2.210000000000e+01  10000                       Chr2, 2.Gen: Gentyp 1
```

Reelle Zahlen, die Null sind, werden als 0 geschrieben. Ansonsten im e-Format mit 12 Nachkommastellen (aus Platzgründen in den Beispielen verkürzt). Beim Einlesen ist natürlich jedes zulässige Format für Double-Werte entsprechend den Konventionen von C zulässig.

Gene ohne Parameter bestehen nur aus dem Gentyp (`activity`, hier 12) und der Pointerkennung, z.B.:

```
12      12345
```

Das zweite Beispiel ist die simulierte Variante des zweiten Beispiels von Abschnitt 2.2. Es unterscheidet sich neben der verkürzten Darstellung der Gene in einigen Kopfdaten und dem Headerparameterelement.

```
   30200  1.048258848543e+01 1 5 2 1 12 2 1  12345
1.086721e+02 2.8000e+00 1.0000e+02 5.0000e+00 5.0000e+00 0 0 0 0 0 0 0 0 0 0 0
0 2 3 0 0 0 0 0 0 0 0 0
    1    7.203019237597e-01 -7.760576578970e+01   12345
    8    1.186834284136e+00   12345
    2    7.359536636808e+00 6.420067142906e+00   12345
```

```
10    2.874823359411e+00   12345
13  19     10000
```

Das Chromosom hat nach Simulation und Bewertung einen Fitnesswert von 10.48258848543 und war dementsprechend unter der Adresse 2/1 einsortiert, als es in die Datei geschrieben wurde. Im Status ist nun auch das Bit `SIMULATED` gesetzt. Das Headerparameterelement enthält nach der Simulation nun die Werte für die 6 Kriterien dieser Anwendung.

4 Literatur

[1] W. Jakob: *HyGLEAM - User Manual*. Technical Paper, KIT, IAI, 2020.
See `HyGLEAM-Manual_V1.0.pdf`

[2] W. Jakob: *MOD-File-Dokumentation*. Internes Arbeitspapier, V1.2, IAI, 2020.
See `MOD-File-Doku_V1.2.pdf`