EVO-File-Dokumentation

GLEAM

Version 1.0

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

Inhalt

Änderungen:

Änderungen gegenüber V1.0:
1.

1 Einführung

Auf die Beschreibung der Verwendung der genetischen Operatoren zur Erzeugung von Nachkommen aus einer Paarung in der Benutzer Dokumentation [1, Abschnitt 3.4] wird verwiesen. Deren Kenntnis ist Voraussetzung zum Verständnis dieser Dokumentation.

Es gibt maximal 10 Operatorengruppen (Konstante `ANZ_PAAR_OPS` in `evo.h`). Jede Operatorgruppe kann aus maximal 16 genetischen Operatoren bestehen (Konstante `ANZ_GEN_OPS` in `go_gsw.c`). Da jede Operatorgruppe bis zu zwei Nachkommen erzeugen kann, sind damit bis zu 20 Nachkommen pro Paarung möglich. Sinnvoll dürften aber geringere Werte im Bereich zwischen 5 und 9 sein.

2 Mutationsraten pro Mutationsoperator

Mutationsoperatoren können mehrfach auf die Gene oder Segmente eines Chromosoms angewendet werden. Die Häufigkeit (Mutationsrate) wird in Abhängigkeit von der Chromosomenlänge ausgewürfelt. Dazu wird jedem der 22 vordefinierten Mutationsoperatoren einer von vier Algorithmen zugeordnet und durch die EVO-Datei parametriert. Gemäß des gewählten Algorithmus wird die Anzahl der Anwendungen eines Mutationsoperators **`anz`** zwischen 1 und `bereich` ausgewürfelt, wobei `bereich` auf die Chromosomenlänge beschränkt wird.

```
Kennung 0 (lin1) : bereich = round (x/a + b);    Min (bereich) = c
                   if (x < a2)
                      anz = 0;
                   else
                      anz = irand (bereich) + 1;
Kennung 1 (lin2) : bereich = round (x/a + b);    Max (bereich) = c
                   if (x < a2)
                      anz = 0;
                   else
                      anz = irand (bereich) + 1;
Kennung 2 (lin3) : if (x < c)
                      bereich = round (x/a  + b);
                      anz = irand (bereich) + 1;
                   else
                      bereich = round (x/a2 + b2);
                      if (bereich < 2)
                         anz = 1;
                      else
                         anz = irand (bereich) + 1;
Kennung 3 (quad) : bereich = round (a*SQR(x) + b*x + c);
                   if (bereich < 2)
                      anz = 1;
                   else
                      anz = irand (bereich) + 1;
```

Die zugehörige Software ist im Modul `go_gsw.c` des `evo`-Packages enthalten. Auf die dortige Dokumentation wird verwiesen.

3 Aufbau einer EVO-Datei

Im Gegensatz zu einer EXP oder TSK-Datei ist eine EVO-Datei formatgebunden. Das bedeutet, dass die Kommentarzeilen einzuhalten sind und es im Grunde auch Leerzeilen sein könnten. Außerdem kann jede Zeile nach den erforderlichen Einträgen beliebigen Text enthalten.

Um die Ausführungswahrscheinlichkeiten der Operatorengruppen und der darin enthaltenen Operatoren an den Evolutionsfortschritt anpassen zu können, sind drei Parametersätze vorgesehen, die je nach Fitness des Elter verwendet werden. Dazu werden im Kopf einer EVO-Datei mit Hilfe zweier Fitnessgrenzen (N_OG1 und N_OG2) drei Fitnessbereiche[1] gebildet. Es gibt entsprechend drei Datensätze für die Parametrierung der Mutationsraten und drei Datensätze für die Konfiguration der Operatorengruppen und die Vorgabe der zugehörigen Wahrscheinlichkeiten.

3.1 EVO-File Kopf

Der Kopf eine EVO-Datei hat folgenden Aufbau:

```
;================================ GLEAM/LESAK ================================
; Parameter f. die Mutations- und Rekombinationsoperatoren.  File: lsk_stnd.evo
; Standard-Einstellungen fuer LESAK                        Stand: 30.06.1998
;============================================================================
GLEAM/AE
40000    60000    Fitnessobergrenzen N_OG1 und N_OG2
22    9           Anz.d.Mut-Ops. (LAST_GO + 1)   Anzahl d. Gen. Operatorgruppen
```

Nach vier Kommentarzeilen folgen die Zeilen mit der Programmkennung `GLEAM/AE` und der Vorgabe der beiden Notenlimits `N_OG1` und `N_OG2`. Diese haben im Beispiel die Werte 40000 und 60000. Daran schließt sich die Anzahl der Mutationsoperatoren und die Anzahl der konfigurierten Operatorgruppen an, hier 22 und 9.

3.2 Parametrierung der Mutationsraten

Danach folgen drei Parametersätze zur Parametrierung der Mutationshäufigkeiten in Abhängigkeit von der Chromosomenlänge. Jeder Parametersatz beginnt mit vier Kommentarzeilen. Z.B.

```
;============================================================================
; Parametersatz 1 von "mut_data" f. Ketten mit schlechter Note (note < N_OG1):
;alg  a         b         c         a2        b2
;============================================================================
```

Die erste Spalte ist die Kennung des Algorithmus zur Berechnung von `bereich` und die weiteren Spalten enthalten die Parameter für den gewählten Algorithmus, siehe Abschnitt 2. Es sind alle Parameter anzugeben, auch wenn der gewählte Algorithmus nur einen Teil davon benötigt.

3.3 Konfigurierung der Operatorgruppen und Vorgabe der Wahrscheinlichkeiten

Nach den drei Parametersätzen zur Parametrierung der Mutationsraten kommen drei Parametersätze zur Zuordnung genetischer Operatoren zu Operatorengruppen samt den jeweiligen Ausführungswahrscheinlichkeiten.

1: Es sei daran erinnert, dass in GLEAM die Fitness den Wertebereich 0 bis 100000 hat, wobei 0 die geringste Fitness darstellt.

Jeder dieser Parametersätze beginnt mit fünf Kommentarzeilen, z.B.:

```
;===========================================================================
; Parametersatz 1 der "go_list" fuer Ketten mit schlechter Note (note < N_OG1):
; p_akt  p_min  eins  op_anz              GenOp- oder Gruppen-Bezeichnung
;   op_list                   p_list                              8.76 Offspring
;===========================================================================
```

In diesem Beispiel enthalten die Kommentarzeilen eine Art Minimaldokumentation und es wurde zusätzlich notiert, wie viel Nachkommen auf Grund der gewählten Wahrscheinlichkeiten im Durchschnitt pro Paarung erzeugt werden.

Danach folgen zwei Zeilen zur Konfiguration einer jeden Operatorengruppe. Deren Anzahl steht Dateikopf. In der ersten Zeile stehen folgende Angaben:

- die Gruppenwahrscheinlichkeit: $0 \leq$ `p_akt` ≤ 1.0
- eine derzeit ungenutzte Angabe `p_min`
- die Angabe, ob ein oder zwei Nachkommen aus der Anwendung der Operatoren dieser Gruppe hervorgehen sollen: 0: zwei Nachkommen, 1: ein Nachkomme
- Die Anzahl der Operatoren dieser Gruppe, mindestens 1.

Die zweite Zeile enthält die Operatorenliste gefolgt von einer Liste der Ausführungswahrscheinlichkeiten je Operator in der ersten Liste.

Drei Beispiele mögen dies illustrieren:

```
  0.8     0.3      0       1                    recomb
    24                            1.0
```

Die Gruppe enthält einen Operator und es werden zwei Nachkommen erzeugt (in der Spalte `eins` steht eine 0). Die Gruppenwahrscheinlichkeit beträgt 80%. In der Operatorenliste der Gruppe steht der Operator 24, was der Rekombination entspricht. Damit ist in GLEAM ein n-Punkt-Crossover gemeint. Die Wahrscheinlichkeit dieses Operators beträgt 100%, was bei einer einelementigen Liste nahe liegt.

Das zweite Beispiel kombiniert das 1-Punkt-Crossover mit zwei Mutationen und die Gruppe wird mit einer Wahrscheinlichkeit von 60% ausgeführt:

```
  0.6     0.15     0       3                    cross_over
    23 1  9                          1.0  0.8  0.5
```

Nach dem Crossover folgt mit 80% Wahrscheinlichkeit eine Parameteränderung am ersten Kind gefolgt von eine Genverschiebung mit 50%. Das 2. Kind bleibt grundsätzlich unverändert.

Das dritte Beispiel mutiert einen Klon des Elter mit einer Gruppenwahrscheinlichkeit von 90% und führt dabei drei Mutationen mit unterschiedlicher Wahrscheinlichkeit aus. Sollte keine zum Zug kommen oder sich zufallsbedingt nichts ändern, wird der Nachkomme gelöscht.

```
  0.9     0.2      1       3                    Neu-Parametrierung
     0   2   4                        0.6   0.7   0.8
```

Die hierbei ausgeführten Mutationen sind:

1. Neuer Parameterwert (0) mit 60 % Wahrscheinlichkeit
2. Neuparametrierung einer Aktion (2) mit 70 % Wahrscheinlichkeit
3. Neuparametrierung aller Aktionen eines Segments (4) mit 80 % Wahrscheinlichkeit

4 Standardoperatoren von GLEAM

4.1 Standardmutationen von GLEAM

Die nachstehende Tabelle gibt den Operatorcode der Standardmutationen wieder (siehe auch `evo.h` im `evo`-Package). Die Mutationen sind in [2, Abschnitt 4.4] beschrieben. Sie können, wie oben ausgeführt, mehrfach auf ein Chromosom angewendet werden.

Operator-code	Beschreibung	Bezeichnung im EVO-File
0	Zufällige Neubestimmung eines zufällig gewählten Genparameters	`par_change_new`
1	Änderung eines oder mehrerer zufällig gewählter Parameterwerte eines Gens	`par_change_rel`
2	Alle Parameter eines zufällig gewählten Gens neu auswürfeln	`act_new_param`
3	Änderung eines oder mehrerer Parameter aller Gene eines zufällig gewählten Segments	`segm_change_rel`
4	Alle Parameter aller Gene eines zufällig gewählten Segments neu auswürfeln	`segm_new_param`
5	Austausch eines zufällig gewählten Gens durch ein zufällig neu bestimmtes	`act_exchange`
6	Einfügung eines neuen Gens an zufällig bestimmter Stelle	`add_new_act`
7	Einfügung der Kopie eines zufällig bestimmten Gens hinter diesem Gen	`double_act`
8	Löschung eines zufällig bestimmten Gens	`delete_act`
9	Verschiebung eines zufällig bestimmten Gens an eine zufällig bestimmte Position	`act_translocation`
10	Ein zufällig neu erstelltes Segment ersetzt ein zufällig bestimmtes Segment	`segm_exchange`
11	Einfügung der Kopie eines zufällig bestimmten Segments hinter dieses Segment	`double_segm`
12	Löschung eines zufällig bestimmten Segments	`delete_segm`
13	Verschiebung eines zufällig bestimmten Segments hinter ein zufällig bestimmtes anderes Segment	`segm_transl`
14	Anordnung der Gene eines zufällig bestimmten Segments in umgekehrter Reihenfolge	`segm_inversion`
15	Ein zufällig bestimmtes Segment wird mit dem nachfolgenden verschmolzen	`integr_nachb_segm`
16	Verschiebung eines zufällig bestimmten Segments vor ein anderes zufällig bestimmtes Segment und Verschmelzung der beiden	`integr_segm`
17	Eine zufällig bestimmte Segmentgrenze wird um eine zufällig bestimmte Anzahl an Genen nach rechts oder links verschoben bis maximal zur nächsten Segmentgrenze	`schieb_segm_grenz`

Operator-code	Beschreibung	Bezeichnung im EVO-File
18	Ein zufällig bestimmtes Segment wird an einer zufällig bestimmten Stelle geteilt, sofern es mindestens zwei Gene hat	`teile_segm`
19	Ein neues zufällig erzeugtes Segment wird hinter einem zufällig bestimmten eingefügt	`add_new_segm`
20	Kleine Änderung eines oder mehrerer zufällig gewählter Parameterwerte eines Gens	`par_change_small`
21	Kleine Änderung eines oder mehrerer Parameter aller Gene eines zufällig gewählten Segments	`segm_change_small`

Tabelle der Standardmutationen von GLEAM

Man beachte, dass die drei Mutationen `integr_nachb_segm`, `schieb_segm_grenz` und `teile_segm` nur die Segmentstruktur verändern und den Phänotyp des Chromosoms somit unverändert lassen. Da unveränderte Nachkommen sofort gelöscht werden, sollten diese Mutationen immer in Verbindung mit anderen Mutationen verwendet werden, es sei denn, dass sie als Mutationen des ersten Nachkommen einer Rekombination oder eines Crossovers angewendet werden.

Die beiden Mutationen `par_change_small` und `segm_change_small` operieren nur über einen Teilbereich des jeweils aktuellen Änderungsintervalls, wobei der Defaultwert ein Tausendstel des jeweils aktuell möglichen Änderungsbereichs beträgt, siehe auch [2, Abschnitt 4.4.3]. Dieser Wert kann im MOD-File durch einen optionalen Eintrag hinter der Angabe der "`akt_roh_erg_werte des (Ext)Simu`„ in Zeile 6 angepasst werden, z.B. auf ein Hundertstel durch Angabe von 0.01.

4.2 Standard-Crossoveroperatoren von GLEAM

In GLEAM sind sieben Standard-Crossover- bzw. Rekombinationsoperatoren implementiert, die alle an den Segmentgrenzen ansetzen. Crossoverpunkte können also nicht innerhalb eines Segmentes liegen. Mit Rekombination werden in GLEAM Crossover-Operatoren bezeichnet, die mehr als 2 Crossoverpunkte haben.

Die nachstehende Tabelle gibt den Operatorcode der Standard-Crossover-Operatoren wieder (siehe auch `evo.h` im `evo`-Package). Die ersten drei Operatoren sind für alle Chromosomentypen geeignet, während die nachfolgenden vier nur Sinn machen, wenn es Reihenfolgerestriktionen bei kombinatorischen Aufgabenstellungen gibt und die Chromosomentypen 2 oder 3 (siehe auch [3] und [2, Abschnitte 4.2.2 und 4.3]) verwendet werden.

Operator-code	Beschreibung	Bezeichnung im EVO-File
22	Austausch zweier zufällig bestimmter Abschnitte der beiden Eltern-Chromosome	`segm_cross_over`
23	1-Punkt-Crossover	`cross_over`
24	n-Punkt-Crossover (Rekombination)	`recomb`
25	Order based crossover als 2-Punkt-Crossover [5, 7]	`OX_XO`
26	Order based crossover als n-Punkt-Crossover [5, 7]	`OX_RECO`
27	Position based crossover als 2-Punkt-Crossover [6, 7]	`PPX_XO`
28	Position based crossover als n-Punkt-Crossover [6, 7]	`PPX_RECO`

Tabelle der Standard-Crossoveroperatoren von GLEAM

Für alle Operatoren gilt, dass die Eltern mindestens zwei Abschnitte haben müssen und bei den beiden OX-Operatoren müssen es sogar mindestens drei sein. Andernfalls werden keine Nachkommen durch die betroffene Operatorengruppe erzeugt.

Die OX- und PPX-Operatoren geben die relative Genreihenfolge der Eltern an die Nachkommen weiter und sind daher nur im Kontext von kombinatorischen Aufgaben mit Reihenfolgebeschränkungen wie z.B. Job-Shop-Scheduling-Problemen sinnvoll. Sie können natürlich auch bei Aufgaben ohne solche Beschränkungen eingesetzt werden, leisten dann aber kaum mehr als die ersten drei Crossover-Operatoren der Tabelle. Die OX-Operatoren haben sich im Gegensatz zu den PPX-Operatoren beim Schedulingaufgaben mit Reihenfolgerestriktionen bewährt [7].

Beide Operatorengruppen berücksichtigen die Segmentierung von GLEAM insofern, als

- dass (ein oder) mehrere Segmente zu den Gensequenzen zusammengefasst werden, die die Grundlage beider Operatoren bilden und
- dass die Segmentstrukturen der Eltern weitervererbt werden.

Bei den Chromosomentypen 1 und 2 kann die Anwendung eines Crossoveroperators dazu führen, dass die Nachkommen zu wenig oder zu viel Gene haben, wobei sich die jeweils fehlenden Gene als überzählige auf dem jeweils anderen Chromosom befinden. Die Überzähligen werden entfernt und an das Chromosomende des jeweils anderen Nachkommen angehängt, wobei sie in das letzte Segment integriert werden.

5 Anwendungsbezogene genetische Operatoren in GLEAM

GLEAM enthält eine Schnittstelle zur Integration anwendungsbezogener genetischer Operatoren. Ihr Operatorcode ist negativ und kann in der Operatorenliste einer Gruppe angegeben werden. Für anwendungsbezogene Mutationen können keine Mutationsraten in der EVO-Datei spezifiziert werden. Die zugehörige SW ist in Form von zwei Funktionen im `appl`-Package im Modul `appl_go.c` enthalten: `do_appl_mut()` für Mutationen und `do_appl_xo()` für Crossover-Operatoren. Erstere wird aufgerufen, wenn der Operatorcode negativ ist und der Eintrag für `eins` für diese Operatorgruppe den Wert 1 hat. Der Operatorcode wird als $|opCode|-1$ an die Funktion weitergegeben, so dass in beiden Funktionen der Wertebereich gültiger Codes bei 0 beginnt.

6 Literatur

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

[2] C. Blume, W. Jakob: *GLEAM - General Learning Evolutionary Algorithm and Method: Ein Evolutionärer Algorithmus und seine Anwendungen*. Karlsruhe: KIT Scientific Publishing, 2009. doi: 10.5445/KSP/1000013553

[3] W. Jakob: *MOD-File-Dokumentation*. Internes Arbeitspapier, V1.2, IAI, 24.7.2020.
See `MOD-File-Doku_V1.2.pdf`

[4] W. Jakob: *Eine neue Methodik zur Erhöhung der Leistungsfähigkeit Evolutionärer Algorithmen durch die Integration lokaler Suchverfahren*. Dissertation, Fak. f. Maschinenbau, Universität Karlsruhe, FZKA 6965, Forschungszentrum Karlsruhe, März 2004. Siehe auch: https://www.iai.kit.edu/~wilfried.jakob/HyGLEAM/

[5] L. Davis (ed): *Handbook of Genetic Algorithms.* V. Nostrand Reinhold, New York (1991)

[6] C. Bierwirth, D.C. Mattfeld, H. Kopfer: *On Permutation Representations for Scheduling Problems*. In: Voigt, H.-M. et al. (eds.): PPSN IV, LNCS 1141, Springer (1996) 310-318

[7] W. Jakob, A. Quinte, K.-U. Stucky, W. Süß: *Fast Multi-objective Scheduling of Jobs to Constrained Resources Using a Hybrid Evolutionary Algorithm*. In: G. Rudolph et al. (eds.): Conf. Proc. of Parallel Problem Solving from Nature X (PPSN 2008), LNCS 5199, Springer-Verlag, Berlin, 2008, S.1031-1040 doi: 10.1007/978-3-540-87700-4_102