

Evaluation des genetischen Algorithmus

Alexander Binsmaier, Manuel Mende, Yaroslav Direy

18. Januar 2017

Inhaltsverzeichnis

1	Multipopulationsalgorithmus	1
1.1	Populationsbehandlung	1
1.2	Genetischer Algorithmus	2
1.3	Kontrollinstanz	3
2	Ausgewählte Testfälle	3
2.1	Gruppierung	3
2.2	Dedizierte Fitnessfunktionen	4
3	Bewertung	4
3.1	Eignung der Fitness-Funktionen	4
3.2	Konvergenz der Populationen	4
3.3	Einfluss der Migrationen	4

1 Multipopulationsalgorithmus

Multipopulations-Algorithmen erlauben das parallele Entwickeln von Populationen. Sie können auf verschiedene Ziele ausgerichtet sein. Ein Austausch der Individuen zwischen Populationen soll zu einer Erhöhung der Diversität führen. Bedingt durch die Notwendigkeit, Populationen als Gesamtheit behandeln und vor allem vermischen zu können, war es notwendig, die zuvor verwendete Implementierung des genetischen Algorithmus anzupassen.

1.1 Populationsbehandlung

Die Verwaltung einer Population wurde in einer Klasse *Population* gekapselt. Sie übernimmt die Erweiterung der Population um neue Chromosome, sowie

die Reduktion auf die vorgegeben Populationsgrößen. Auch die Initialisierung der Chromosome wird von der Populationsklasse übernommen. Die Reduktion der Chromosomen-Anzahl erfolgt durch aussortieren der „schwächsten“ Individuen. Sie zeichnen sich durch die schlechteste Fitness innerhalb der Population aus.

1.2 Genetischer Algorithmus

Die erste Version des genetischen Algorithmus verfügte lediglich über eine Hauptroutine, die die zyklische Evolution der Chromosome vornahm. Um mehrere Algorithmen verwenden und eine Interaktion gewährleisten zu können, musste eine schrittweise Umsetzung der Evolution integriert werden. Aus Gründen der Übersichtlichkeit und Wartbarkeit wurde eine neue Klasse **GenericGA** angelegt. Diese weist zwei Vorteile gegenüber der ersten Implementierung auf. Die Algorithmus-Klasse ist lediglich für die Umsetzung der genetischen Operationen zuständig, sämtliche die Population betreffende Operationen wie Initialisierung und Reduktion werden in der Populations-Klasse behandelt. Zudem kann die Evolution Schrittweise von einer übergeordneten Kontrollinstanz durchgeführt werden. Damit ergibt sich die Flexibilität, verschiedene Instanzen zu erzeugen, die Populationen zu mischen, und erneut einige Iterationen zu durchlaufen.

Die schrittweise Evolution wird durch die Funktion

`runEpoch(population)`

gewährleistet. Sie modifiziert Chromosome der übergebenen Population. Dazu werden die Funktionen-Handles der **GenericGA**-Instanz verwendet, die durch den Konstruktor gesetzt werden müssen. Als problematisch erweist sich die Ermittlung der neuen Fitness-Werte. An sich ist es sinnvoll, vor jedem Selektionsprozess die Fitnessfunktionen zu berechnen. Damit wäre für einen nächsten Durchlauf die alte Fitness für die Rekombination verfügbar. Leider kann das im Falle von Migrationen zu Problemen führen, da hier Chromosome aus anderen Populationen übernommen werden. Weil die verschiedenen Populationen durch unterschiedliche Fitnessfunktionen geprägt werden, ist nach einer Migration vor einer erneuten Rekombination eine erneute Fitnessberechnung durchzuführen. Die Schnittstelle `runEpoch` setzt deshalb Populationen mit validen Fitnessinformationen voraus. Dazu kann die Funktion

`computeFitness(population)`

einer **GenericGA**-Instanz verwendet werden. Sie weist jedem Chromosom die durch die Fitnessfunktion des genetischen Algorithmus bedingte Fitness zu.

Sollen mehrere Evolutionsepochen ohne Migration durchlaufen werden, kann die Schnittstelle

```
runEpochs(population, iterations)
```

verwendet werden. Sie setzt keine validen Fitness-Informationen voraus.

1.3 Kontrollinstanz

Die Kontrollinstanz koordiniert die Ausführung der einzelnen genetischen Algorithmen. Die Initialisierung erfolgt lediglich mit der vorgesehenen Größe der Population. Die Klasse `MultiPopulationGA` enthält die Funktionen

```
addPopulation(fitness_handle) sowie  
runMPGA(migrations_rate, epochs_between_migrations, migration_cycles, policy)
```

Erstere dient dem konfigurieren der genetischen Algorithmen – hier kann eine besondere Fitnessfunktion übergeben werden – die zweite Funktion dient der Evolution der Populationen. Die Abarbeitung kann durch die Parameter eingestellt werden. Die Migrationsrate entspricht der Anzahl der Chromosome, die in jeder Population getauscht werden sollen. Zudem kann die Anzahl Epochen zwischen Migrationsvorgängen und die Anzahl an Migrationsvorgängen festgelegt werden. Der Parameter `policy` dient dem Wechsel zwischen Ring-, Nachbar- und uneingeschränkter Migration.

2 Ausgewählte Testfälle

Zur Definition der einzelnen Populationen für den genetischen Algorithmus wurden Äquivalenzklassen der Testfälle analytisch hergeleitet.

2.1 Gruppierung

Die Anzahl der Äquivalenzklassen für die gesuchten Testfälle wird durch Kombination von Fahrzeugposition im Verhältnis zur Parklücke (left/right) sowie von Fahrzeugausrichtung (up/down/left/right) bestimmt. Somit ergeben sich acht Gruppen der gesuchten Testfälle.

2.2 Dedizierte Fitnessfunktionen

Für jede Äquivalenzklasse wurde eine Fitnessfunktion so definiert, dass die Testfälle mit gesuchten Eigenschaften innerhalb einer Population eine höhere Wertung erhalten. Somit wird der Selektionsdruck vom genetischen Algorithmus in eine für gegebenen Population gewünschte Richtung geleitet. Die Testfälle, die von den gewünschten Eigenschaften stark abweichen werden dagegen mit einem schlechten Wert der Fitnessfunktion bestraft und folglich aussortiert.

3 Bewertung

Die Ergebnisse vom genetischen Algorithmus mit mehreren Populationen weisen eine hohe Heterogenität im Vergleich zu einem einfachen genetischen Ansatz. Für jede Population wird der Selektionsdruck und somit auch die Konvergenz der gesuchten Testfälle mittels angepassten Fitnessfunktionen gesteuert.

3.1 Eignung der Fitness-Funktionen

Die Ergebnisse vom genetischen Algorithmus für acht Populationen zeigen, dass die jeweiligen Fitnessfunktionen die Evolution der Testfälle in die gewünschte Richtung treiben.

3.2 Konvergenz der Populationen

Die Konvergenz der Testfälle in der Populationen wird durch die verwendete Fitnessfunktion gesteuert. Durch eine hohe Bewertung der Testfälle mit gewünschten vorprogrammierten Eigenschaften bzw. durch Abwertung der unpassenden Testfälle konvergieren die Chromosome mit steigender Anzahl der Epochen zunehmend.

3.3 Einfluss der Migrationen

Für die Migration der Chromosome zwischen den Populationen wurden die 'ring', 'unrestricted' sowie die 'neighbour' Methode verwendet. Wie aus dem Vergleich der Ergebnisse ohne Migration ersichtlich hat Einsatz der genannten Migrationsmethoden positiv zur Erhöhung der Heterogenität der Testfälle innerhalb der Populationen beigetragen.

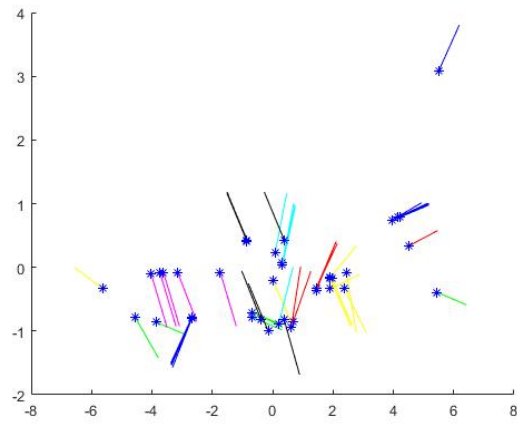


Abbildung 1: Visualisierung der Testfälle: MPGA ohne Migration

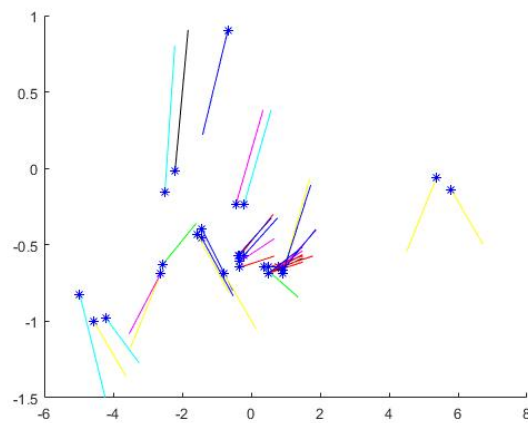


Abbildung 2: Visualisierung der Testfälle: MPGA Ring Migration

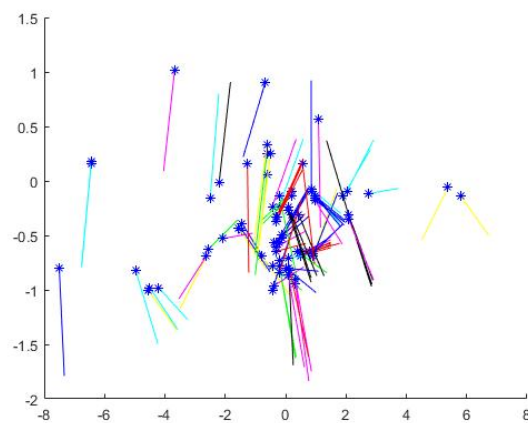


Abbildung 3: Visualisierung der Testfälle: MPGA Unrestricted Migration

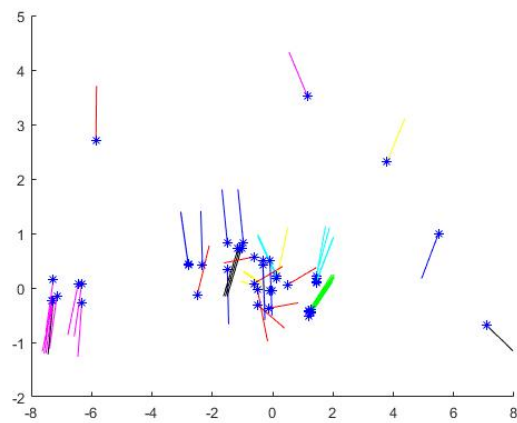


Abbildung 4: Visualisierung der Testfälle: MPGA Neighbour Migration