Evaluation des genetischen Algorithmus

Alexander Binsmaier, Manuel Mende, Yaroslav Direy

16. Januar 2017

Inhaltsverzeichnis

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 betreffenden 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 weißt 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

2.1 Gruppierung

2.2 Dedizierte Fitnessfunktionen

3 Bewertung

3.1 Eignung der Fitness-Funktionen

3.2 Konvergenz der Populationen

3.3 Erfolg der Migrationen