Projektarbeiten

1. Allgemeine Erklärung zum DGPF-Projekt

Das Distributed Genetic Programming Framework nutzt Genetische Algorithmen, um automatisch Programme für bestimmte Problemstellungen zu finden (Genetic Programming). Die Programme sind zurzeit in einer Assembler-ähnlichen Form gehalten und werden interpretiert. Das Hauptziel des Projektes ist es, einfache Algorithmen zur verteilten Problemlösung in (Sensor-)Netzwerken zu finden. Der für die Evolution von solchen Algorithmen notwendige Rechenaufwand wird über ein Netzwerk von Computern mit Hilfe einer Peer-To-Peer Technik verteilt. Diese implementiert einen „Island-Hopping“ genannten Mechanismus. Während das System eine Population von Programmen heranzieht, können viele Parameter der Evolution angepasst werden.

Da das DGPF-System OpenSource und Freeware ist, werden auch die Arbeiten der Studenten unter LGPL-Lizenz als Teil des Projekts veröffentlicht. Mit Hilfe dieser Lizenz wird es anderen Entwicklern ermöglicht, das DGPF-Projekt in ihre Anwendungen einzubinden oder auch für andere Einsatzgebiete Genetischer Algorithmen zu nutzen. Um den internationalen Einsatz des DGPF weiter zu vereinfachen, sind die Projektarbeiten in Englisch auszufertigen. Programmkode muss vollständig dokumentiert werden, um es anderen/späteren Entwicklern zu ermöglichen, nahtlos an die Arbeit anzuknüpfen.

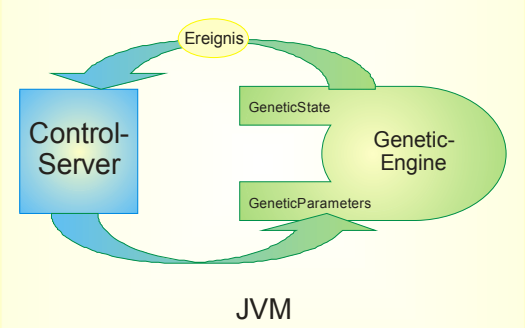
Zwischen den Aufgaben gibt es Nahtstellen, die die Zusammenarbeit und Kommunikation der beiden Teams erfordern. Auch ist es teilweise möglich, durch das Nutzen gemeinsamer Komponenten die Arbeit zu vereinfachen.

2. Aufgabe: Managementsystem / GUI (2-3 Studenten)

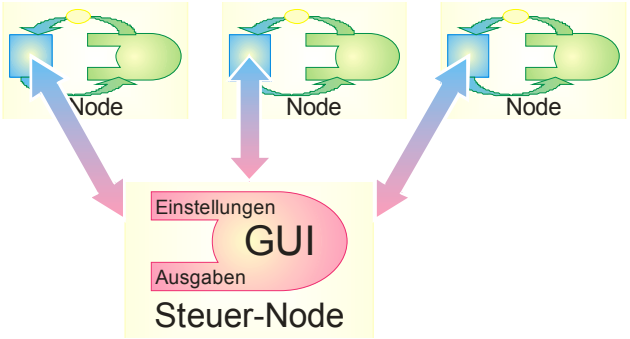
Es ist ein erweiterbares Managementsystem mit Benutzeroberfläche für das DGPF-System zu entwerfen und mit Hilfe der Programmiersprache Java zu implementieren. Das zu erstellende Managementsystem hat zwei Hauptaufgaben: die vom System

erzeugten statistischen Daten darzustellen (sowie auszuwerten und aufzuzeichnen) und dem Benutzer zu ermöglichen, das System zu steuern.

Die Oberfläche muss dem Fakt Rechnung tragen, dass es sich bei dem DGPF-System um eine verteilte P2P-Anwendung handelt: Die GUI muss die Steuerinformation über das Netzwerk zu den Genetic-Engines auf den einzelnen Knoten übertragen. Die Knoten senden dafür Zustandsinformationen (z.B. Statistiken) zum Managementsystem. Die Genetic-Engine an sich besitzt lediglich ein System, um Ereignisse zu erzeugen und eine Schnittstelle, mit der dynamisch Einstellungen vorgenommen werden können. Die Datenkommunikation mit dem Knoten auf dem, das Managementsystem läuft, muss als Teil der Aufgabe realisiert werden und könnte wie in den Grafiken 1 und 2 ablaufen. Besonders wichtig ist, dass die Datenkommunikation zwischen Managementsystem und Genetic-Engine asynchron abläuft, also dass die Genetic-Engine nicht durch zu langsame Kommunikation mit dem Managementsystem gebremst wird.



Grafik 1: Kontrolle der Genetic-Engine und Erhalten ihrer statistischen Daten



Grafik 2: Zusammenführung der Steuerung und Auswertung zur GUI

Die GUI soll es dem Benutzer jeweils ermöglichen die Informationen des Gesamtsystems sowie der einzelnen P2P-Knoten auszuwerten. Ebenso soll die Steuerung sowohl des Gesamtsystems als auch der einzelnen Knoten möglich sein.

Zur Datendarstellung sollen konfigurierbare Diagramme genutzt werden. Dafür ist es sinnvoll, im Internet nach vordefinierten Komponenten zu suchen. Allgemein dürfen in das Projekt nur Komponenten integriert werden, die auch unter der LGPL Lizenz stehen. Der Benutzer soll einen guten Eindruck vom Fortschritt der Evolution erhalten und nachvollziehen können, wie sich seine Einstellungen auf das System auswirken – dafür ist eine verständliche und klar strukturierte Aufbereitung der Informationen notwendig.

Das DGPF-Framework gliedert sich in mehrere Ebenen. Die Genetic-Engine an sich ermöglicht den Einsatz beliebiger Genetischer Algorithmen für beliebige Problemdomänen. Darüber befindet sich die Ebene der Genetischen Programmierung – diese erlaubt es, Algorithmen in einer Assembler-ähnlichen Notation zu erzeugen. Während also Diagramme genutzt werden können, um die allgemeinen statistischen Ausgaben der Genetic-Engine darzustellen, benötigt man zur optischen Aufbereitung der darüber liegenden Ebenen eine austauschbare Komponente. Die Spezifikation und Implementierung einer Schnittstelle für diese Darstellungskomponente in die GUI sowie die Implementierung einer solchen

Darstellungskomponente für die genetisch erzeugen Programme des Systems ist ebenfalls Teil der Aufgabe.

Der Benutzer soll Steuerinformationen per Hand mit der GUI setzen und an das System übertragen können.
Das Managementsystem soll zu einem späteren Zeitpunkt so erweitert werden, dass Fitnessfunktionen ausgewählt und eine genetische Evolution gestartet werden kann. Wenn dies auch nicht Teil der Aufgabe ist, so soll zumindest diese zukünftige Entwicklung bereits im Entwurf und der Spezifikation bedacht werden.
Das Managementsystem soll bereits die Entwicklungen der Studenten, die die Aufgabe „*Finden von günstigen Rückkopplungsschleifen für die Genetische Programmierung*“ bearbeiten erwarten und eine spätere Integration vorbereiten.

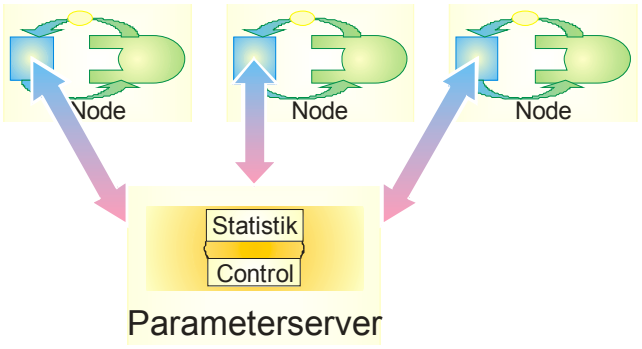
3. Aufgabe: Finden von günstigen Rückkopplungsschleifen für die Genetische Programmierung (1-2 Studenten)

Die Genetic-Engine des DGPF produziert am Ende jeder Generation eine Menge von statistischen Informationen. Nachdem dies geschehen ist, werden die Parameter für die nächste Generation (z.B. Populationsgröße, Mutationsrate, Emigrationsrate, Selektionsalgorithms usw.) von einer Einstellungsschnittstelle (Klasse „`GeneticParameters`“) eingelesen und angewendet.
Dadurch ist es einerseits möglich, günstige statische Parameterkonfigurationen zu entwerfen. Die weitaus interessantere Variante ist jedoch das Entwickeln von dynamischen Strategien, so genannten Rückkopplungsschleifen. Diese nutzen die statistischen Informationen der bisherigen Generationen, um möglichst günstige Parameter für die folgenden Generationen zu finden. Man kann auch von einer Art Meta-Evolution sprechen: die Genetischen Algorithmen werden über ihre Laufzeit hinweg langsam optimiert.

Ein sehr simples Beispiel für eine solche Rückkopplungsschleife wäre es zu untersuchen, welcher genetische Operator die besten Nachkommen produziert. Wenn also z.B. die Mutation über einen gewissen Zeitraum hinweg bessere Nachkommen als das Crossover erzeugt, wäre es nur logisch, den Anteil durch Mutation zu erzeugender Nachkommen gegenüber dem von Crossover zu erhöhen.
Es stellt sich auch die Frage, wie schnell unser System reagieren soll – ist es sinnvoll, die Parameter nach jeder Generation neu zu adjustieren, oder sollte man dies nur gelegentlich tun, und dafür mehr statistische Daten nutzen?
Zum Bearbeiten dieser Aufgabe gehört es, verschiedene Rückkopplungsschleifen zu entwickeln und zu experimentell testen. Dazu ist es notwendig, entsprechende Experimente zu erstellen und mehrmals zu wiederholen. Eine umfassende Liste mit Vorschlägen für Rückkopplungsschleifen existiert bereits – es ist jedoch durchaus auch erwünscht, eigene Ideen zu integrieren.
Es wäre zudem nützlich, die gemachten Beobachtungen mit Hilfe stochastischer und anderer mathematischer Methoden zu prüfen oder sogar vorherzusagen.

Durch die P2P-Struktur des Frameworks kommt noch ein weiterer sehr interessanter Aspekt hinzu: Evolutionsstrategien können nicht nur lokal sondern auch global

arbeiten. Eine zentrale Instanz, die regelmäßig von den einzelnen Knoten statistische Informationen erhält, könnte z.B. erkennen, dass ein Knoten besonders gute Individuen hervorbringt. Sie könnte die Emigrationsrate dieses Knotens erhöhen und seine Immigrationsrate erniedrigen. Bei den anderen Knoten im Netzwerk wird stattdessen die Immigrationsrate erhöht und Emigrationsrate erniedrigt – so würden sich die besseren Individuen schneller global durchsetzen.
Hier wäre es nötig, eine Serverkomponente zur Genetic-Engine hinzuzufügen (wie in Grafik 1 in Aufgabe „*Managementsystem/GUI*“ vorgestellt), die die Datenkommunikation mit der zentralen Instanz zur Parameterbestimmung („Parameterserver“ in Grafik 3) regelt. Das System muss asynchron genug sein, so dass weder der Parameterserver gebremst werden, wenn z.B. ein Knoten langsamer läuft als alle anderen.



Grafik 3: Zentraler Parameterserver

Neben Entwurf und Implementierung eines solchen Parameterservers ist das Entwickeln von zentralen Evolutionsstrategien Teil der Aufgabe. Dazu gehört es auch, es zu ermöglichen, dass sowohl globale als auch lokale Strategien arbeiten. Die globale Instanz könnte beispielsweise nur ein paar Parameter festlegen, die dann die Einstellungen der lokalen Strategien überschreiben. Bei den restlichen Parametern soll jedoch die lokale Strategie verfolgt werden.
Dadurch ist es auch möglich, dass sich das System selbstständig weiter optimiert, wenn die Kommunikation mit dem Parameterserver ausfällt.

Es sind die Arbeiten der Studenten der Aufgabe „*Managementsystem/GUI*“ bereits in der Konzeption des Systems zu beachten: es muss später möglich sein, dass die GUI mit ihren Einstellungen wiederum diejenigen des zentralen Parameterservers überschreiben kann. Auch sollten Schnittstellen vorgesehen werden, die die Ausgabe von Informationen über die Aktionen des Parameterservers an die GUI ermöglichen würden – die Implementierung ist jedoch nicht Teil der Aufgabe.

4. Vorschlag zur Zusammenarbeit

Beide Teams benötigen eine Server-Erweiterung der P2P-Knoten des DGPFs. Es wäre ein guter Ansatz, diese zuerst gemeinsam zu entwickeln, bevor mit den anderen Arbeiten begonnen wird. Würde man den Server als eine Art von Subscriber-System realisieren, so käme das einer späteren Koexistenz der beiden Entwicklungen sehr entgegen.