University of Kassel, Chair of Distributed Systems, Wilhelmshöher Allee 73, D-34121 Kassel, Hesse, Germany

Projektarbeiten

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 kann dafür über ein Netzwerk von Computern verteilt werden. Dazu können verschiedene Verteilungsmechanismen wie Client/Server (C/S), Peer-To-Peer P2P) oder eine C/S-P2P-Hybridform genutzt werden. Während das System eine Population von Programmen heranzieht, können viele Parameter der Evolution angepasst werden. Weiterhin ist das gesamte System modular aufgebaut, so dass es nicht an die Genetische Programmierung gebunden, sondern für beliebige Problemdomänen wie z.B. numerische Regression, Web-Service-Composition oder Sudoku-Lösen eingesetzt werden kann. Dazu stehen nicht nur Genetische Algorithmen, sondern auch andere heuristische Suchverfahren wie Hill Climbing und Simulated Annealing zur Verfügung, welche auch parallel und gemischt betrieben werden können.

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. 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.

In vorangegangenen Projekten haben Studenten bereits eine vielseitige graphische Oberfläche für das Projekt erstellt, ein neues Suchverfahren in das Projekt integriert

und ein paar Performance-Tests durchgeführt sowie neue Problemdomänen beispielhaft angebunden.

1. Aufgabe: Deployment auf einem Cluster (1-2 Studenten)

Wie bereits in den allgemeinen Erklärungen erwähnt ist das DGPF-Projekt selbst eine skalierbare, verteilte Anwendung. Probleme, die man mit diesem Framework lösen kann, stellen verschiedenste Anforderungen. Die zum Auswerten einer möglichen Lösung benötigten Simulationen sind oftmals rechenaufwendig, in anderen Fällen benötigt man sehr große Populationen um überhaupt gute Lösungen finden zu können – manchmal tritt auch beides gemeinsam auf.

In dieser Aufgabe soll das DGP Framework auf den Cluster des Universitätsrechenzentrums deployed werden. Es werden die verschiedenen zur Verfügung stehenden Verteilungsmechanismen getestet und die Suchverfahren (GA, Simulated Annealing, Hill Climbing) werden einzeln und parallel im Zusammenspiel miteinander betrieben.

Nach den ersten Tests mit dem Projekt wird eine detaillierte (englischsprachige) Anleitung für das Deployment des DGPFs auf einem Cluster erstellt. Anforderungen wie das Vorhandensein einer JVM, die belegten Ports und der voraussichtlich Ressourcenverbrauch werden darin erläutert.

Ziel des Projekts ist es, optimale Konfigurationen für unterschiedliche Problemklassen zu finden. Diese werden aus Messwerten (Geschwindigkeit der Lösungsfindung bzw. Konvergenzverhalten der Fitness im Allgemeinen) abgeleitet.

Dazu sind neben dem verteilten Testen der bereits vorhandenen Beispiele aus der Genetischen Programmierung auch die Experimente von [1] nachzuvollziehen.

Mit dem erforschten Wissen wird die Anleitung erweitert. Einen Ratgeber soll dem Anwender helfen, aus den Eigenschaften seiner Problemdomäne ein geeignetes Verteilungskonzept (Suchverfahren, Verteilungsmechanismus, benötigte Rechner) zu erstellen.

Meilensteine:

M1: Anleitung für das Deployment des DGPFs auf einem Cluster
M2: Anleitung für das Verteilen eines Problems

[1] Marion Riedel: "Parallele Genetische Algorithmen mit Anwendungen", Diplomarbeit TU Chemnitz, http://archiv.tu-chemnitz.de/pub/2002/0134/index.html

2. Aufgabe: Erweitern der GUI um Projektmanagement (2 Studenten)

In einem vorangegangenen Projekt haben Studenten bereits eine vielseitige graphische Oberfläche für das DGPF-Projekt erstellt. Mit dieser ist es möglich, sich auf entfernte Knoten zu verbinden und dort eine Suche zu starten, stoppen und zu überwachen.
Eine Suche wird durch die Problemdomäne, die Fitnessfunktionen, die zu verwendenden Suchverfahren, ihre Verteilungsart und die Start- sowie Halteparameter festgelegt. Die GUI erfordert aktuell, dass diese Definition bereits als (kompilierte Java-Klassen in einem) jar-Archiv abgelegt ist.

In diesem Projekt soll die vorhandene GUI um Editoren für die Erstellung einer Suchdefinition erweitert werden. Der für Fitnessfunktionen oder die Spezifikation von Problemdomänen notwendige Java-Kode soll in einem einfachen Editor bearbeitet und in Dateien oder ggf. einer Datenbank abgelegt werden können.
Der Benutzer kann sich dann die gewünschte Suchdefinition mit Hilfe neuer Dialoge zusammenstellen und das benötigte jar-Archiv wird automatisch im temporären Verzeichnis erstellt.

Die bereits vorhandenen Komponenten gestatten es, beiliebig viele Rechenressourcen für die Lösung eines Problems einzubinden und die Problemdomänen völlig frei zu definieren. Mit Abschluss dieses Projekts erreicht das DGPF jedoch einen Reifegrad, der es selbst Standardanwendern mit geringen Vorkenntnissen ermöglicht, komplexe Probleme automatisiert zu lösen.

Meilensteine:

M1: Dialoge zur Auswahl von Verteilungsart und Suchalgorithmus sowie Anbindung des JavaC- oder Eclipse-Compilers zum Erzeugen eines entsprechenden jar-Archivs
M2: kleiner Java-Editor, der für Fitnessfunktions-Erstellung genutzt wird
M3: Nutzung des Editors für Problemdomänenspezifikation

3. Aufgabe: WSC-Challenge 2007 (1-2 Studenten)

Im Jahr 2006 ist es dem Team Bleul-Weise der Professur Verteilte Systeme an der Uni Kassel gelungen, den Web Service Challenge ([2], [3], [4]) zu gewinnen. Dem Wettbewerb lagen folgende Prinzipien zugrunde:

1. Webservices werden durch die Beschreibungssprache WSDL beschrieben, sie haben Ein- und Ausgabeparameter.
2. Semantische Konzepte sind in Baumform einer Ontologie beschrieben.
3. Die Ein- und Ausgabeparameter von Webservices genügen jeweils einem semantischen Konzept.
4. Services können hintereinander ausgeführt werden, in dem die Ausgabeparameter (plus eine Menge bereits bekannter Konzepte) des zuerst ausgeführten Service die Eingabeparameter des nächsten auszuführenden Service bilden.

Das System hat Zugriff auf die Ontologie und eine Menge von Webservice-Beschreibungen. Dem System wird eine Menge bekannter Konzepte und eine Menge gesuchter Konzepte übergeben. Gesucht sind alle möglichen (minimal langen) Hintereinanderausführungen der bekannten Services, mit denen die unbekannten Konzepte aus den bekannten bestimmt werden können.

Nachdem die im Jahr 2006 gestellte Aufgabe geschickt auf zwei (heuristische) Tiefensuchen reduziert werden konnte, ist das Gesamtproblem an sich NP-vollständig. Aus diesem Grund empfiehlt sich der Einsatz randomisierter Verfahren, wie sie vom DGPF-Projekt zur Verfügung gestellt werden. Tatsächlich existiert bereits eine prototypische, einfache Implementation als Beispiel.

Ziel dieses Projektes ist es, auf Basis der Siegertechnologie des letzten Jahres und mit Hilfe der Beispielimplementation der Service-Komposition des DGPFs eine wettkampffähige Lösung für die Web Service Challenge 2007 zu entwickeln.

Dazu müssen neue Heuristiken und Fitness-Funktionen für die Web-Service Komposition entwickelt werden. Günstige Werte für die Parameter der randomisierten Suchverfahren sind zu wählen und es sind Zeitmessungen sowie Optimierungen durchzuführen

Meilensteine:

M1: Tests mit dem vorhandenen Kompositionsalgorithmus und dem DGPF-Beispiel
M2: Entwicklung neuer Fitness-Funktionen, Optimierungen am Gesamtsystem, Performanzmessungen
M3: Fertiges, wettkampffähiges System mit voller Dokumentation und vollständigen Messprotokollen

[2] http://insel.flp.cs.tu-berlin.de/wsc06/ (Ergebnisse)
[3] http://ws-challenge.georgetown.edu/ws-challenge/Participants.htm (Teilnehmerseite)
[4] http://www.vs.uni-kassel.de/ADDO/ (Projekt Seite)

4. Aufgabe: Finden von günstigen Rückkopplungsschleifen für die Genetische Programmierung (1 Student)

Die Suchverfahren des DGPF produzieren 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) von einer Einstellungsschnittstelle 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 Operator bei Genetischen Algorithmen 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 verteilte 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.

Meilensteine:

M1: lokale Analysen und Adaptionsverfahren
M2: globale Analysen und Adaptionsverfahren