Entwurfsbeschreibung des HELIOS-Projekts

swp-helios

19. Mai 2014

Inhaltsverzeichnis

1 Allgemeines

Ziel des Projekts ist es, einen neuen Planungsalgorithmus zu entwickeln, um zu beliebigen Metriken einen möglichst effizienten Plan zu finden. Auch ist geplant eine browserbasierte GUI zu entwickeln, um LIMES einfacher nutzbar zu machen.

2 Produktübersicht

Das Gesamtprodukt besteht aus mehreren Bestandteilen: Ein auf LIMES basierender Benchmark, der zum Testen und Vergleichen von Planungsalgorithmen benutzt wird, um den entwickelten Algorithmus mit anderen Algorithmen zu vergleichen und mögliche Schwachstellen zu finden. Zweiter Bestandteil ist der Planungsalgorithmus, der zur Ermittlung eines effizienten Planes innerhalb LIMES benutzt wird. Der letzte Teil ist eine browserbasierte Benutzeroberfläche für Endanwender, sodass Endanwender die Berechnungen einfach, ohne das Programm selbst auf dem Endbenutzer-Computer zu installieren, ausführen können.

3 Grundsätzliche Struktur-und Entwurfsprinzipien

3.1 Planungsalgorithmus

Der Planungsalgorithmus wird als ein genetischer Algorithmus implementiert, der eine lokale Suche auf dem Raum aller möglichen und korrekten Pläne für die gegebene Metrik ausführt. Dabei wird die schon von LIMES benutzte Bibliothek JGAP[1] verwendet.

3.2 Browserbasierte GUI

Über eine Benutzeroberfläche auf der Webseite sollen Linkspecs und benötigte Parameter angegeben werden können. Das Ergebnis einer Link Discovery wird zum Download zur Verfügung gestellt.

4 Struktur-und Entwurfsprinzipien einzelner Pakete

4.1 Genetischer Algorithmus

Die Startpopulation soll zufällig sein und wird dadurch erzeugt, dass alle Felder des Tupels (siehe Datenmodell) zufällig gewählt werden. Mutationen

[1] `http://jgap.sourceforge.net/`

und Kreuzungen werden auf diesen Feldern ausgeführt.

4.2 Fitness-Funktion

Die Bewertung der Pläne erfolgt gemäß der Funktion `1 + 1/time`. `time` bezeichnet die Zeit, die das Auführen des Plans auf zufälligen Teilmengen der Ausgansmengen S und T benötigt. Diese Funktion genügt dem Interface JGAPs, indem ihr Wert mindestens 1 ist und sie mit weniger zeitaufwändigen Plänen wächst.

4.3 Browserbasierte GUI

Mittels Apache Wicket wird auf unserer Webseite dynamisch ein HTML-Formular erstellt. Es soll dem Benutzer die Wahl gegeben sein, entweder eine LIMES-Konfigurationsdatei hochzuladen oder die benötigten Daten direkt in vorgegebene Felder einzutragen. Zusätzlich wird es die Möglichkeit geben, direkt einen gewünschten Plan anzugeben.
Nach Ausführung der Link Discovery, wird man das Ergebnis herunterladen können. Außerdem wird der benutzte Plan in einem Graph angegeben, wofür wir mit graphviz ein Bild serverseitig generieren. Die benötigte Zeit und Anzahl der Tripel werden genannt.
Für die grafische Gestaltung wird das Front-End Bootstrap verwendet, das einfach zu bedienende und visuell ansprechende Oberflächenbuttons und Textflächen bietet und einfach mit JavaScript und Apache Wicket verknüpft werden kann. Damit ist eine geordnete und modern anmutende Oberfläche zu visualisieren. Eigene Linkspezifikationen können in einem separaten Popup eingegeben werden. Es gibt einen Filebrowser zur Auswahl von XML-Files und Buttons zur Berechnung des Ergebnisses und zum Download der Tripel, dazu eine einfache Algorithmus-Wahlliste, in der zwischen dem kanonischen und unserem genetischen Algorithmus gewählt werden kann. Die user-spezifizierten Parameter der Linkspec werden als Strings an unser Programm gesendet und dann weiterbenutzt.

5 Datenmodell

5.1 Interne Repräsentation des Plans

Die Pläne werden als Tupel der Entscheidungen repräsentiert, wobei es für jedes `AND` 5 Möglichkeiten gibt, dieses umzusetzen. Diese sind:

0. Bestimmen der Ergebnisse beider Seiten, dann schneiden
1. (2.) Bestimmen des Ergebnisses der linken (rechten) Seite, dann Filtern nach dem Ergebnis der anderen Seite

3. (4.) Bestimmen des Ergebnisses der linken (rechten) Seite, dann Einschränken der Quell- und Zielmenge auf die noch mögliche Teilmenge, Bestimmen des Ergebnisses der rechten (linken) Seite auf diesen Mengen, schneiden

5.2 Externe Repräsentation des Plans

Im Folgenden seien $M_0, M_1, \ldots$ Mappings, $(m_0, \theta_0), (m_1, \theta_1), \ldots$ Metriken und $(S_0, T_0), (S_1, T_1), \ldots$ Quell- und Zielmengen.

M_0 - Endmapping
$(m_0, \theta_0), (m_1, \theta_1), \ldots$ - nach in-order traversal nummeriert
(S_0, T_0) - Ausgangsmengen

Sprache:

$$
\begin{aligned}
exp :=\ & M_a := RUN, (S_b, T_b), (m_c, \theta_c) \\
\mid\ & M_a := FILTER, M_b, (m_c, \theta_c) \\
\mid\ & (S_a, T_a) := RESTRICT, M_b \\
\mid\ & M_a := (UNION|INTERSECT|DIFF|XOR), M_b, M_c \\
\mid\ & exp; exp
\end{aligned}
$$

5.3 Überführung der internen zur externen Repräsentation

$exp[a := b]$ bedeutet, dass in dem Ausdruck exp jedes Vorkommen von a durch b ersetzt wird.
Die Funktion P, die den Plan erstellt ist:

$P((m_a, \theta_a), run) = M_0 := RUN, (S_0, T_o), (m_a, \theta_a)$
$P((m_a, \theta_a), M_b) = M_0 := FILTER, M_b, (m_a, \theta_a)$

O.B.d.A.: Rekursive Aufrufe benutzen nur (S_0, T_0), M_0 und (S_i, T_i), M_i mit $i \geq 3$, was durch Umbenennen erreichbar wird.

$$
\begin{aligned}
&P(AND(L_1, L_2), x, 0, i_1, \ldots, i_a, j_1, \ldots, j_b) = \\
&\qquad P(L_1, x, i_1, \ldots, i_a)[M_0 := M_1]; \\
&\qquad P(L_2, x, j_1, \ldots, j_b)[M_0 := M_2]; \\
&\qquad M_0 := INTERSECT, M_1, M_2
\end{aligned}
$$

$$
\begin{aligned}
&P(AND(L_1, L_2), x, 1, i_1, \ldots, i_a, j_1, \ldots, j_b) = \\
&\qquad P(L_1, x, i_1, \ldots, i_a)[M_0 := M_1]; \\
&\qquad P(L_2, M_1, j_1, \ldots, j_b)
\end{aligned}
$$

$$
\begin{aligned}
&P(AND(L_1, L_2), x, 2, i_1, \ldots, i_a, j_1, \ldots, j_b) = \\
&\qquad P(L_2, x, i_1, \ldots, i_a)[M_0 := M_1]; \\
&\qquad P(L_1, M_1, j_1, \ldots, j_b)
\end{aligned}
$$

$$
\begin{aligned}
&P(AND(L_1, L_2), x, 3, i_1, \ldots, i_a, j_1, \ldots, j_b) = \\
&\qquad P(L_1, x, i_1, \ldots, i_a)[M_0 := M_1]; \\
&\qquad (S_1, T_1) := RESTRICT, M_1; \\
&\qquad P(L_2, run, j_1, \ldots, j_b)[M_0 := M_1][(S_0, T_0) := (S_1, T_1)]; \\
&\qquad M_0 := INTERSECT, M_1, M_2
\end{aligned}
$$

$$
\begin{aligned}
&P(AND(L_1, L_2), x, 3, i_1, \ldots, i_a, j_1, \ldots, j_b) = \\
&\qquad P(L_2, x, j_1, \ldots, j_b)[M_0 := M_1]; \\
&\qquad (S_1, T_1) := RESTRICT, M_1; \\
&\qquad P(L_1, run, i_1, \ldots, i_a)[M_0 := M_1][(S_0, T_0) := (S_1, T_1)]; \\
&\qquad M_0 := INTERSECT, M_1, M_2
\end{aligned}
$$

6 Testkonzept

Die Teststrategie für unser Projekt ist an die Anforderungen angepasst. Dabei verfolgen wir einen top-down Ansatz. Dort werden Haupt- vor Detailfunktionen getestet. Dies hat den Vorteil, dass Kernfunktionen, wie zum Beispiel der Algorithmus, mit höherer Wahrscheinlichkeit ein zufriedenstellendes Niveau erreichen. Dabei sollen sowohl statische als auch dynamische Tests zum Einsatz kommen. Es ist sinnvoll, gemeinsam am Code zu arbeiten um Fehler frühzeitig zu erkennen. Als flexible kleine Gruppe kann natürlich auch situationsbedingt getestet werden, um so notwendige Funktionen schnell auszureifen. Dies kann vorallem auf die Entwicklertests (Programmiertests) zutreffen. Die Dokumentation bei diesen Tests beschränkt sich auf einen Testvorfallbericht. Nach Abschluss der grundlegenden Arbeiten liegt das Hauptaugenmerk des Testkonzepts besonders auf Benchmark, Oberflächentest und Abnahmetest.

6.1 Benchmark

Das Benchmark, welches im Vorprojekt realisiert wurde, bildet den Kern des Testkonzepts. Es ermöglicht eine Beurteilung der Algorithmen bei verschiedenen Anforderungen. Dadurch ist ein Vergleich möglich, welcher uns Hinweise auf die Leistungsfähigkeit unseres Algorithmus gibt. Dieser muss sich gegen die vorhandenen Planer, den kanonischen Planer und HELIOS, behaupten. Das Benchmarking kann einige Zeit beanspruchen. Es sind die Benchmarkspezifikationen und die Benchmarkergebnisse zu dokumentieren um Vergleiche besser ziehen zu können. Da das Benchmark teil des Performanztests ist sollte ebenfalls die Speicherauslastung im Auge behalten werden. Zu Beginn der Benchmarkingphase wird gleichzeitig ein Abnahmetest des Vorprojekts durchgeführt um die Aussagekraft des Benchmarks sicherzustellen.

6.2 Oberflächentest

Unser Projekt wird als Webanwendung zur Verfügung gestellt. Dies bedeutet, dass Anwender von verschiedensten Engeräten mittels Browser auf unserer Projekt zugreifen werden. Dabei muss einiges beachtet werden damit sich die Anwendung dem Benutzer in allen Bereichen gleich darstellt. Die Tests sollten in den beliebtesten Browsern durchgeführt werden und auf Darstellung und die Funktionstüchtigkeit von Standardfunktionalitäten achten. Auch sollte während der Entwicklung auf Standardkonformität geachtet werden. Die Benutzbarkeit muss ebenfalls überprüft werden um Verständlichkeit und Anordnung der Informationen zu optimieren.

6.3 Abnahmetest

Zu guter Letzt steht natürlich der Abnahmetest an. Hier wird überprüft ob unsere Arbeit dem Anspruch des Auftraggebers genügt. Wichtig hierbei ist das Verhalten der Software bei spezifizierten Handlungen des Benutzers. Diese Teststufe sollte besondere Aufmerksamkeit genießen um Nachbesserungen wegen Beanstandungen zu vermeiden.

7 Glossar

Apache Wicket

Apache Wicket ist ein komponentenbasiertes objektorientiertes Web-Framework für Java. Hauptgedanke des Frameworks ist die Trennung von in Java geschriebener Logik und der Darstellung in HTML bzw. CSS.

Benchmark

Ein Benchmark ist eine Vergleichende Analyse. In unserem Fall sollen Algorithmen verglichen werden. Dabei ist besonders die Performance entscheidender Bewertungsfaktor.

GUI

Ein Graphical User Interface, (dt. eine grafische Benutzeroberfläche) ist eine Schnittstelle zwischen Benutzer und Programm, die nicht vollständig auf Text basiert, sondern auch grafische Darstellungen (z.B. in unserem Fall von Graphen) zur Darstellung von Daten verwenden kann.

Genetischer Algorithmus

Heurisitk zur Lösung von Entscheidungsproblemen.Genetische Algorithmen dienen der Lösung von Optimierungsaufgaben.

JGAP

JGAP (Java Genetic Algorithms and Genetic Programming Package) ist ein Java Framework zur Umsetzung von Genetischen Algorithmen.

Link Discovery

Das Auffinden von Links zwischen Wissensbasen im Semantic Web bezeichnet man als Link Discovery.

Linkspezifikation

Beschreibung der Ähnlichkeitskriterien zweier verschiedener RDF-Ressourcen zum Zweck der Verknüpfung.

Planungsalgorithmus

Ein Planungsalgorithmus ist ein Algorithmus, der aus einem Abhängigkeitsgraph von Arbeitsschritten eine die Abhängigkeiten nicht verletzende lineare Abfolge der Arbeitsschritte erstellt.