Dokumentation

ANTS ON CRACK

Der Ameisenalgorithmus als Lösung

für das Problem des Handlungsreisenden

Jonathan Wiens, Nico Smeenk, Benjamin Pfeiffer, Christian Gebauer, Sebastian Große

Berlin, der 25.10.2011

Kontakt der Verantwortlichen:

Name	**Kontakt**
Jonathan Wiens	jonathan.wiens@gmail.com
Nico Smeenk	nicosmeenk@gmx.de
Benjamin Pfeiffer	benjamin.pfeiffer.berlin@gmx.de
Christian Gebauer	christian.gebauer.stud@gmail.com
Sebastian Große	p.servus.servus@gmail.com

Infos:

Dies ist die Dokumentation zu dem Projekt “ANTS ON CRACK”, ein Projekt im Rahmen des Studiengangs Informatik im dritten Semester der Hochschule für Wirtschaft und Recht im Fach “Software Engineering I” unter der Aufsicht von Prof. Dr. Dagmar Monett Díaz.

Titel: Einer für alle und alle für die Ameise

Kurzfassung

Dies ist die Dokumentation zu dem Projekt “ANTS ON CRACK” unter der Aufsicht von Prof. Dr. Dagmar Monett Díaz.

Die Aufgabenstellung des Projektes besteht darin das Traveling Salesman Problem (TSP) mithilfe des Ameisen-Algorithmus zu lösen. Dabei sollen Städte und Touren visualisiert und die Resultate ausgegeben werden. Im Vordergrund steht die gezielte Anwendung erlernter XP-Techniken (eXtreme Programming) aus den Veranstaltungen. Hierzu gehören unter anderem Pair Programming, die Nutzung von Story Cards, ständige Refaktorisierungen, sowie schnelle Codereviews. Der folgende Text beschreibt die theoretische Aufgabenstellung und die praktische Umsetzung mit dem Blickpunkt gerichtet auf Extreme Programming.

Inhaltsverzeichnis

Teil I. Einführung

1 Traveling Salesman Problem

Das Traveling Salesman Problem (TSP, zu deutsch Problem des Handlungsreisenden) ist ein Problem in der kombinatorischen Optimierung der theoretischen Informatik und erfüllt die Vorraussetzungen zur nichtdeterministisch polynomielle Vollständigkeit (NP-Vollständigkeit). NP-vollständig bedeutet, dass es wahrscheinlich keinen effizienten Algortihmus zum Problem gibt. Es bleiben Verfahren, die sich einer guten Lösung annähern, wie es bei probabilistischen, heuristischen oder Approximationsverahren der Fall ist.[1]

Das Problem lässt sich dadurch beschreiben, dass eine Liste von Städten mit Koordinaten gegeben ist und die kürzeste Route, in der jede Stadt einmal besucht wird und die Startstadt gleich der Endstadt ist, gesucht wird. Um die Problemstellung des TSP graphisch veranschaulichen zu können, werden die Städte als Knoten und die Wege zwischen zwei Städten als Kanten repräsentiert.[2]

2 Ameisenalgorithmus

In der Informatik ist der Ameisenalgorithmus eine Metaheuristik, die sich auf das Schwarmverhalten von Ameisen auf Futtersuche stützt.

In der Natur wandern Ameisen zunächst zufällig, bis sie Futter gefunden haben und kehren anschließend zu ihrer Kolonie zurück, wobei sie Pheromonspuren hinterlassen. Sobald andere Ameisen einen Pfad mit Pheromonspuren finden, wandern sie nicht mehr zufällig, sondern folgen dem Pfad und verstärken ihn dadurch, bis sie schließlich das Futter finden. Währenddessen verdunsten die Pheromonspuren und verringern somit die Attraktivität des Pfades. Je länger eine Ameise für einen Weg braucht, desto mehr verdunsten die Pheromone auf dem Pfad.

Wenn ein Pfad kurz ist, wird dieser öfters genutzt und die Pheromonspur wird stärker. Somit werden kurze Wege den längeren Wegen vorgezogen. Wenn eine Ameise einen attraktiven, bzw. kurzen Weg zwischen der Kolonie und der Futterquelle findet, ist es wahrscheinlicher, dass andere Ameisen ebenso diesen Pfad folgen und somit positives Feedback produzieren, bis alle Ameisen einem einzigen Weg folgen oder in anderen Worten eine Ameisenstraße bilden.

Im Allgemeinen kann man die Regeln zur Bildung einer Ameisenstraße in drei einfache Schritte zusammenfassen (siehe Abbildung 1):

1. Die erste Ameise findet die Futterquelle (F) über einen zufälligen Weg, kommt zurück zum Nest (N) und hinterlässt eine Pheromonspur

2. Die Ameisen folgen einen der 4 möglichen Wegen. Die Verstärkung der Wege macht den kürzesten Weg am attraktivsten

3. Die Ameisen folgen den kürzesten Weg, die Pheromonspuren der längeren Wege verdunsten.

Der Ameisenalgorithmus soll dieses Verhalten in der Natur simulieren, so dass damit eine gute Lösung für einen Handlungsreisenden erreicht wird.

[1] H.W. Lang, *NP-Vollständigkeit*,
http://www.inf.fh-flensburg.de/lang/algorithmen/np/npvollst.htm
[2] Sariel Har-Peled, *Traveling Salesman Problem*,
http://www.valis.cs.uiuc.edu/~sariel/research/CG/applets/tsp/TspAlg.html

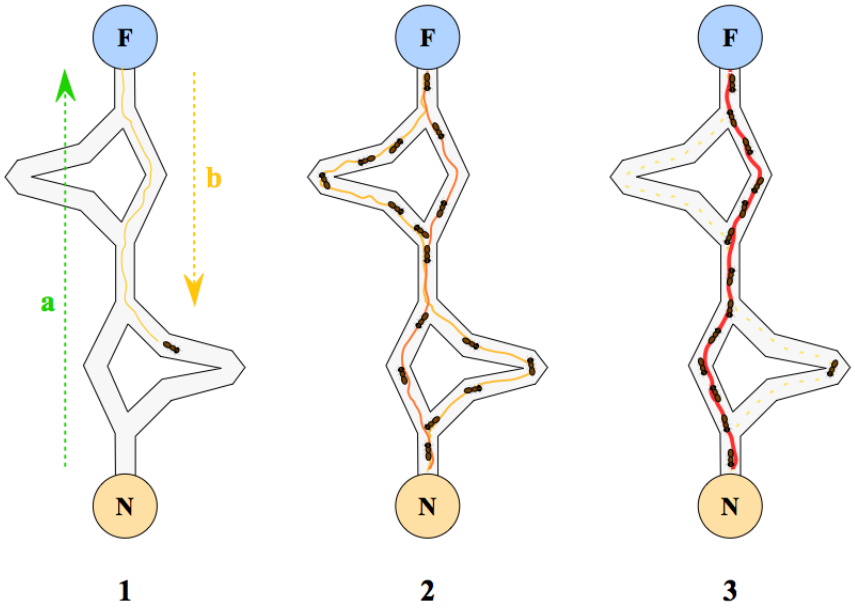

Abbildung 1: Veranschaulichung des Verhaltens von Ameisen in der Natur.[3]

Teil II. Agile Softwareentwicklung

Agile Softwareentwicklung ist ein Trend in der Informatik, der eine Menge von Methoden zur Entwicklung von Software umfasst, die auf iterative und inkrementative Entwicklung basieren. Im Manifesto der agilen Softwareentwicklung[4] wird ein Wertesystem aufgestellt, welche Auslebung eine bessere Möglichkeit zum Schreiben von Software bieten soll.

Insgesamt wurde das Projekt in zwei Iterationen aufgeteilt; in der ersten Iteration wurde ein Grundgerüst mit GUI implementiert, in der zweiten Iteration die hinterliegende Logik.

Die folgenden Punkte spiegeln die Werte im Manifesto für agile Softwareentwicklung wieder und sind offiziell Teile der Extreme Programming Regeln[5]. Sie werden theoretisch erklärt und im praktischen Bezug vom Projekt erläutert.

3 Umgebung des Projektes

Dieser Abschnitt behandelt die allgemeine Festlegungen, die am Anfang des Projektes festgelegt werden und nur in Ausnahmefällen geändert werden sollten.

3.1 Rollen- und Aufgabenverteilung

Rollen sind in einem Projekt laut Extreme Programming immer klar definiert. Jeder hat Zugang zu den Rolleninformationen und weiß somit welches Teammitglied für welchen Aufgabenteil zuständig ist. Somit können alle Beteiligten sich auf ihren Bereich spezialisieren und mit minimalisierten Unterbrechungen arbeiten.

[3] Vittorio Maniezzo, Luca Maria Gambardella, Fabio de Luigi, *5. Ant Colony Optimization, http://www.idsia.ch/~luca/aco2004.pdf*

[4] Kent Beck, *Manifesto for Agile Software Development*, http://www.agilemanifesto.org/

[5] Extreme Programming ist eine der weltweit meistgenutzte agile methodistische Entwicklungsart. Für Informationen: Don Wells, *The Rules of Extreme Programming*, http://www.extremeprogramming.org/rules.html

In diesem programmier- und dokumentierschwerem Projekt wurde jedem Mitglied, mit einer Ausnahme, eine Teilrolle als Programmierer und Dokumentierer gegeben. Zusätzlich gab es noch die Einzelrollen des Projektleiters[6], des Architekten, Serveradministrators und die der Qualitätssicherung (siehe Abbildung 2).

Abbildung 2: Die Teammitglieder mit Vornamen und Rollenvergabe.

3.2 Festlegung einer Entwicklungsumgebung

Zu Beginn eines Softwareprojektes sollte die Programmiersprache oder bei Bedarf auch mehrere Programmiersprachen festgelegt werden, in der das Projekt implementiert werden soll. Dabei spielt es nicht nur eine wichtige Rolle, welche Sprachen sich für die Aufgaben eignen, sondern welche Sprache von den Programmierern am Besten beherrscht wird.

Die Wahl der Programmiersprache hat auch einen Einfluss, welche Möglichkeiten für eine Daten- und Klassenstruktur zur Verfügung stehen. Dabei muss ein Kompromiss zwischen Redundanz und Geschwindigkeit gefunden werden. Um den Vorgaben von XP treu zu bleiben, wurde für dieses Projekt die unter den Mitgliedern meist genutzte Sprache Java genutzt. Dadurch wurde der Mehraufwand für die Mitglieder reduziert, der in jeder anderen Sprache aufgrund von Unkenntnis der Sprachumgebung, aufgekommen wäre.

3.3 Code Conventions

Ebenfalls vor Beginn sollten gewisse Standards für das Schreiben von Code festgelegt werden. Es bietet sich an, dafür allgemein bekannte, bzw. übliche Standards zu verwenden, da diese in der Regel intuitiv anwendbar sind.

Code Conventions dienen dazu, den Code übersichtlich, gut lesbar und leicht anpassbar zu machen. Es wird dabei unter anderem die Art der Einrückung, die Klammersetzung und die Namensgebung beschrieben. Auch werden hier die Standards zur Kommentierung des Codes festgehalten.[7]

4 Verwaltung des Projektes

Die folgenden Punkte beziehen sich auf die XP-Methoden, die die spezifische Organisation und Verwaltung des Projektes ermöglichen.

4.1 Story Cards

Das Planning Game[8] wird zu Beginn einer jeden Iteration des Projektes gespielt, um die Anforderungen zu definieren und deren Zeitaufwand, Priorität u.Ä. abzuschätzen. Zu jeder Anforderung wird eine

[6] Die Rolle des Projektleiters besteht unter anderem darin einen Überblick aller Prozesse zu haben, eine ständigen Arbeitsfluss aufrecht zu erhalten und, im Fall dieses Projektes auf Grund ihrer Größe, Teile der Aufgaben zu übernehmen. Für mehr Informationen: Realcomm, *Job Description – Project Manager*, http://www.realcomm.com/RC07-Placement/Library/Emp/RCPS_job_description_project_manager_1a.pdf

[7] Ian Mitchell, *Coding Standards for Java*, http://www.xp.co.nz/Coding_Standards_for_Java.htm

[8] James Shore, *The Art of Agile Development: The Planning Game*, http://jamesshore.com/Agile-Book/the_planning_game.html

Story Card geschrieben, die die zugehörigen Abschätzungen beinhaltet. Wichtig dabei ist, dass jede Story Card bestimmten Spezifikationen entspricht[9]; auf jeder Karte stehen die gleichen Felder (siehe Abbildung 3).

Titel:	#10 GUI - Handbuch
Datum:	08.09.2011
Autor:	Sebastian
Beschreibung:	▪ kurze Anleitung des Programmes
Art:	fachliche Anforderung
Geschätzter Aufwand:	4 h
Tatsächlicher Aufwand:	2 h
Datum der Erledigung:	15.10.2011
Entwickler:	Benjamin, Jonathan
Priorität:	1

Abbildung 3: Eine Story Card bzw. User Story aus dem Projekt "ANTS ON CRACK"

4.2 Daily Stand Up Meeting

In dem Daily Stand Up Meeting (auf deutsch die tägliche Besprechung im Stehen) soll jeder Teilnehmer in einer Runde drei Fragen beantworten:

1. Was habe ich seit dem letzten Stand-Up Meeting getan?
2. Welche Probleme traten dabei auf?
3. Was werde ich als nächstes tun?

Diese Besprechung sollte im Stehen durchgeführt werden, um längere Sitzungen zu vermeiden.[10]

5 Teamarbeit und Kommunikation

5.1 Kommunikation

Programmierer, die gemeinsam an einem Projektes arbeiten, sollten auch zusammen in einem Büro sitzen, um immer schnell und effektiv miteinander kommunizieren zu können. Mitarbeiter und Programmierer, die nicht in das Projekt involviert sind, sollten dagegen nach Möglichkeit räumlich getrennt untergebracht werden, um unnötige Störquellen und Ablenkung zu vermeiden.

5.2 Pair Programming

Beim Pair Programming programmieren zwei Programmierer gemeinsam an einem Computer. Dabei schreibt einer den Code und der andere sieht Ersterem über die Schulter und steht diesem beratend, bzw. korrigierend zur Seite; die Rollen der Programmierer sollten regelmäßig getauscht werden.

Dadurch kann eine erhöhte Qualität des Codes bewerkstelligt werden, die im Falle von zwei seperaten Programmierern schlechter wäre.[11]

[9] Don Wells, *User Stories*, http://www.extremeprogramming.org/rules/userstories.html
[10] Wichtig zu vermeken ist, dass keine speziellen Diskusionen induziert werden dürfen, diese sollten später mit den betreffenden Personen besprochen werdenl; in dem Daily Stand Up Meeting soll lediglich eine Bestandsaufnahme stattfinden. Don Wells, *Daily Stand Up Meeting*, http://www.extremeprogramming.org/rules/standupmeeting.html
[11] Don Wells, *Pair Programming*, http://www.extremeprogramming.org/rules/pair.html

Teil III. Praktische Umsetzung des Programms

In den folgenden Texten wird die praktische Umsetzung des Ameisenalgorithmus auf TSP-Probleme dargestellt.[12]

6 Abstraktion Ameisenalgorithmus

6.1 Einheiten

Folgende Einheiten wurden im Programm für die Umsetzung des Ameisenalgorithmus' festgelegt.

6.1.1 Ameise

Eine Ameise läuft für sich alleine eine Route ab. Dabei wird nach dem Ameisenalgorithmus die Wahrscheinlichkeit eines Weges zwischen zwei Knoten bestimmt. Für die Bewertung eines Weges siehe Teil 6.2. Ein Weg kann nicht zwei Mal gelaufen werden. Die Länge ihrer Route wird gespeichert und falls es die kürzeste ist als kürzeste Route gespeichert.

6.1.2 Iteration

In einer Iteration laufen n Ameisen als Menge eine Route ab. Nach einer Iteration wird ein Pheromonupdate ausgeführt (siehe Teil 6.2.1).

6.2 Wegbewertung

Die Wegbewertung setzt sich aus zwei zu berechnenden Faktoren zusammen, der Pheromonbewertung, sowie der lokalen Information.

6.2.1 Pheromonupdate

Das Pheromonupdate bestimmt die Stärke der Pheromone, die von den Ameisen auf den Wegen abgelassen werden, sowie die Verdunstung die Eintritt, nach dem eine Iteration durchlaufen wurde.

6.2.2 Lokale Information

Die lokale Information bestimmt für eine Ameise wie attraktiv ein bestimmter Weg auf Grund der Weglänge ist.

7 Programmablauf

Das folgende Diagramm zeigt den gewünschten Programmablauf (siehe Abbildung 4):

[12] Aufgrund Dokumentenlängenbegrenzung kann nicht ins Detail gegangen werden. Für mehr Informationen über das Projekt und Implementierungsdetails siehe http://lisa-debian-server.dyndns.org/mediawiki/index.php/Projekt:_TSP-Ameisenalgorithmus

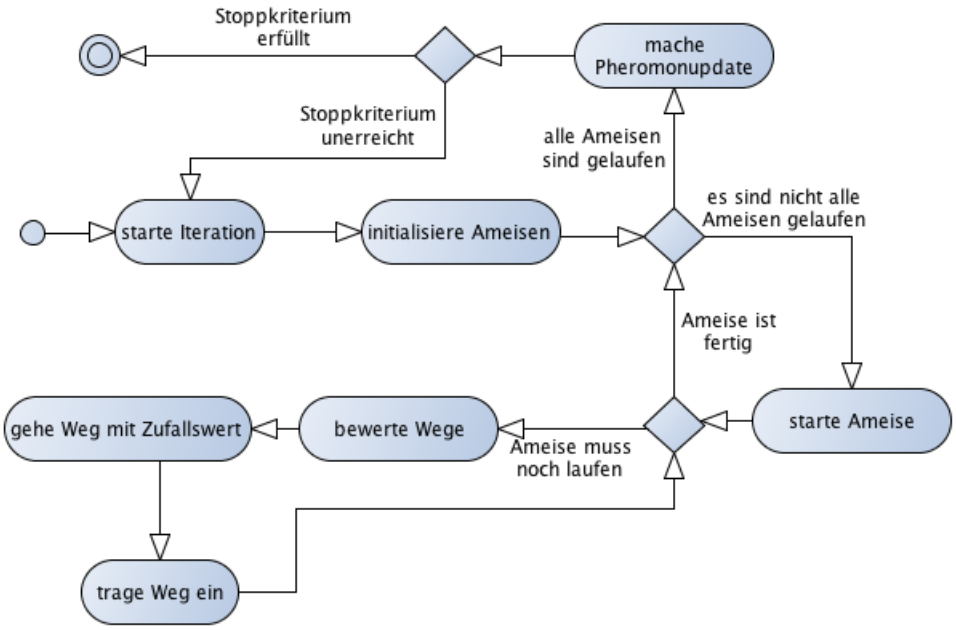

Abbildung 4: Der Programmablauf anhand eines Flussdiagramms dargestellt.

7.1 Stoppkriterien

Der Algorithmus könnte theoretisch unendlich lang laufen, deswegen werden bestimmte Stopp-Kriterien definiert:

- Durchlaufen einer bestimmten Anzahl von Iterationen
- Erreichen eines bestimmten Schwellwertes, z.B. kürzeste Tour $<$ x
- Erreichen des optimalen Weges

Teil IV. Fazit und Ausblick

8 Fazit

Das Projekt wurde erfolgreich abgeschlossen, erfüllt alle Kundenanforderungen und kann ohne Schwierigkeiten erweitert und optimiert werden, da eine modulare, objekt-orientierte Architektur mittels XP-Techniken erreicht wurde.

Abschließend kann man sagen, dass die erlenten XP-Techniken sich nicht nur positiv auf die Programmentwicklung ausgewirkt hat, sondern auch die Motivation der Mitarbeiter durch Erhöhung der Effizienz jedes Einzelnen Mitgliedes erhöht hat. Die Philosophie, die hinter Extreme Programming steckt, wurde mit Freuden angenommen und ausgeführt.

9 Ausblick

Da das Projekt nur ca. zweite Monate zur Entwicklung Zeit hatte und neben dem Projekt andere Projekte anstanden, konnte keine ausführliche Prozessoptimierung und Qualitätssicherung durchgeführt werden. Deshalb sollte man im Zukunftsblick im jeden Fall eine Optimierung des Algorithmus' hinsichtlich der projektspezifischen Implementierung haben, sowie eine statistische Auswertung von guten, bzw. besten Lösungen nach Zeiten erstellen.

10 Bibliografie

Autor/en, Titel	**Referenz**
H. W. Lang, NP-Vollständigkeit	http://www.inf.fh-flensburg.de/lang/algorithmen/np/npvollst.htm
Sariel Har-Peled, Traveling Salesman Problem	http://www.valis.cs.uiuc.edu/~sariel/research/CG/applets/tsp/TspAlg.html
Vittorio Maniezzo, Luca Maria Gambardella, Fabio de Luigi, 5. Ant Colony Optimization	http://www.idsia.ch/~luca/aco2004.pdf
Kent Beck, Manifesto for Agile Software Development	http://www.agilemanifesto.org/
Don Wells, The Rules of Extreme Programming	http://www.extremeprogramming.org/rules.html
Realcomm, Job Description – Project Manager	http://www.realcomm.com/RC07-Placement/Library/Emp/RCPS_job_description_project_manager_1a.pdf
Ian Mitchell, Coding Standards for Java	http://www.xp.co.nz/Coding_Standards_for_Java.htm
James Shore, The Art of Agile Development: The Planning Game	http://jamesshore.com/Agile-Book/the_planning_game.html
Don Wells, User Stories	http://www.extremeprogramming.org/rules/userstories.html
Don Wells, Daily Stand Up Meeting	http://www.extremeprogramming.org/rules/standupmeeting.html
Don Wells, Pair Programming	http://www.extremeprogramming.org/rules/pair.html