

pb.net:
Konzeption, Design und Implementierung eines Frameworks
zur Entwicklung von Heuristiken
für Kombinatorische Probleme

Inauguraldissertation
zur
Erlangung des Doktorgrads
der
Wirtschafts- und Sozialwissenschaftlichen Fakultät
der Universität zu Köln

2009

vorgelegt
von
Dipl.-Wirt.-Inf. Paul Bartodziej
aus
Oppeln

Referent:	Prof. Dr. Dr. Ulrich Derigs
Korreferent:	Prof. Dr. Werner Mellis
Tag der Promotion:	3. Juli 2009

Wirtschaftsinformatik und Operations Research

Band 12

Paul Bartodziej

pb.net:

Konzeption, Design und Implementierung eines Frameworks
zur Entwicklung von Heuristiken für Kombinatorische Probleme

D 38 (Diss. Universität Köln)

Shaker Verlag
Aachen 2009

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Zugl.: Köln, Univ., Diss., 2009

Copyright Shaker Verlag 2009

Alle Rechte, auch das des auszugsweisen Nachdruckes, der auszugsweisen oder vollständigen Wiedergabe, der Speicherung in Datenverarbeitungsanlagen und der Übersetzung, vorbehalten.

Printed in Germany.

ISBN 978-3-8322-8274-5

ISSN 1433-8521

Shaker Verlag GmbH • Postfach 101818 • 52018 Aachen

Telefon: 02407 / 95 96 - 0 • Telefax: 02407 / 95 96 - 9

Internet: www.shaker.de • E-Mail: info@shaker.de

Drei Schüler waren in einem undurchdringlichen Dickicht unterwegs. Sie suchten und suchten nach dem Heimweg, doch sie fanden ihn nicht. Nun trafen sie dort ihren Meister.

Sie liefen auf ihn zu und fragten: „Ach, Meister! Schön, dass du da bist. Du kamst uns hier heraus-helfen. Wo ist der Weg? Wir finden ihn nicht.“

„Ich kann euch nicht den Weg heraus zeigen, sondern nur den, der noch tiefer ins Dickicht führt“, antwortete er.

Anekdote aus Fernost

Danksagung

Allen voran danke ich meinem Doktorvater Herrn Prof. Dr. Dr. Ulrich Derigs für die vorzügliche Betreuung. Seine Ratschläge und Anregungen sind mir auf meinem Weg eine wertvolle Orientierungshilfe gewesen.

Darüber hinaus bedanke ich mich bei Herrn Prof. Dr. Werner Mellis für die Übernahme des Zweitgutachtens.

Ebenfalls gilt mein Dank meinen Kollegen, ehemaligen Kollegen und Freunden Thorsten Akkerman, Stefan Bell, Dr. Felix Bomsdorf, Sascha Dahl, Thomas Döhmer, Dr. Stefan Ems, Stefan Friederichs, Gisela Hafner, Olaf Jenal, Jochen Kuritz, Dominik Malcherek, Dr. Shehab Marzban, Michael Piesche, Simon Schäfer, Ralf Schnickmann, Daniel Weber und Ulrich Vogel für die ausgezeichnete Arbeitsatmosphäre am Seminar für Wirtschaftsinformatik und Operations Research. Während der Entstehung dieser Arbeit habe ich auf die freundliche Unterstützung eines jeden von Ihnen bauen können.

Zu guter Letzt möchte ich meiner Familie herzlich für ihr Vertrauen und ihren Rückhalt danken. Ganz besonders danke ich meinem Vater, der mir in Sachen Zielstrebigkeit und Ausdauer, die mir bei der Fertigstellung dieser Arbeit zugutegekommen sind, stets ein großes Vorbild ist.

I. Inhaltsverzeichnis

I. Inhaltsverzeichnis.....	v
II. Abbildungsverzeichnis	xi
III. Tabellenverzeichnis.....	xi
IV. Abkürzungsverzeichnis.....	xii
1 EINLEITUNG: MOTIVATION UND ZIELE DES FRAMEWORKS	1
1.1 Grundlagen	1
1.2 Motivation.....	4
1.2.1 Auslastung der gegebenen Rechenressourcen	4
1.2.2 Implementierung von Optimierungs-Algorithmen durch mehrere Personen.....	5
1.2.3 Entwicklung heuristischer Ideen für neuartige Probleme	7
1.3 Ziele.....	9
1.3.1 Effiziente Parallelisierung	9
1.3.2 Vereinfachte Entwicklung durch modulare Solver-Architektur.....	9
1.3.2.1 Arbeitsteilung.....	10
1.3.2.2 Weitere Vorteile des modularen Ansatzes	11
1.3.3 Rapid Prototyping	11
1.3.4 Wiederverwendung	12
1.4 Weiterer Aufbau der Arbeit	13
2 GRUNDLAGEN: ALGORITHMIK UND TECHNOLOGISCHE BASIS.....	15
2.1 Metaheuristiken.....	15
2.1.1 Grundlagen von Metaheuristiken	15
2.1.2 Bekannte Metaheuristiken.....	18
2.1.2.1 Simulated Annealing.....	19
2.1.2.2 Tabu-Suche	22

2.1.2.3	Genetische Algorithmen.....	25
2.2	Frameworks zur Entwicklung von Metaheuristiken.....	28
2.2.1	Traditionelle Konzepte.....	29
2.2.2	Das Konzept von pb.net.....	31
2.3	Das Microsoft .NET-Framework.....	32
2.3.1	Grundlagen des Microsoft .NET-Frameworks	33
2.3.1.1	Historischer Vergleich.....	33
2.3.1.2	Hauptziele des .NET-Frameworks.....	36
2.3.1.3	Problematik der Interoperabilität.....	39
2.3.1.4	Aufbau des .NET-Frameworks.....	42
2.3.2	Nutzung der Vorteile des Microsoft .NET-Frameworks.....	46
3	GRUNDLAGEN: PARALLELISIERUNGSKONZEPTE	51
3.1	Motivation.....	51
3.2	Grundlegende Parallelisierungskonzepte.....	51
3.2.1	Parallele Architektur	52
3.2.2	Granularität.....	53
3.2.3	Synchronität.....	53
3.2.4	Parallelismus.....	54
3.2.5	Programmiermodell	55
3.3	Parallelisierungskonzepte bei der Lokalen Suche	57
3.3.1	Walk-orientierte Klassifikationen	57
3.3.2	Klassifikation bezüglich Kontrolle und Kommunikation.....	62
3.4	Kommunikations- und Memory-Konzepte bei Multiple Walks	66
3.4.1	Allgemeine Kommunikationsaspekte.....	66
3.4.2	Unabhängige Walks	68
3.4.3	Synchrone kooperative Walks.....	69
3.4.4	Parallele Memory-Konzepte	70

3.4.5	Performanznachteile bei synchronen kooperativen Walks.....	72
3.4.6	Asynchrone kooperative Walks.....	73
3.4.7	Globales Memory bei asynchronen kooperativen Walks	74
3.4.8	Beispiel 1: Steuerung der Suche durch Globales Memory.....	76
3.4.8.1	Ausgangsproblem.....	77
3.4.8.2	Lösungsverfahren.....	79
3.4.9	Beispiel 2: Steuerung der Suche ohne Globales Memory	81
3.4.10	Aktuelle Trends bei parallelen Algorithmen.....	86
3.5	Parallelisierungskonzepte von pb.net.....	89
4	KONZEPTION: OPTIMIERUNGSLOGIK IM FRAMEWORK.....	91
4.1	Grundlagen der Optimierungslogik	91
4.2	Schema zur Steuerung von Subheuristiken	93
4.2.1	Hyperheuristik-Ansatz.....	93
4.2.1.1	Grundlagen.....	93
4.2.1.2	Anwendungsbeispiel	98
4.2.2	Vergleich mit verwandten Metaheuristiken.....	101
4.2.2.1	Iterated Local Search und Variable Neighborhood Search.....	101
4.2.2.2	Large Neighborhood Search.....	104
4.2.3	Adaptierung des Hyperheuristik-Konzepts im Framework.....	106
4.3	Schema zur Diversifikation.....	108
4.3.1	Abstract Local Search.....	108
4.3.2	Anwendungsbeispiel zu Abstract Local Search.....	113
4.3.3	Squeaky Wheel Optimization.....	115
4.3.4	Anwendungsbeispiel zu Squeaky Wheel Optimization	119
4.3.5	Synthese: Adaptierung der ALS/SWO-Idee in pb.net	120
4.4	Kombination von Parallelisierung mit Subheuristik-Steuerung und Diversifikation	121

5	DESIGN: AUFBAU DES FRAMEWORKS.....	125
5.1	Schnittstelle für Lösungslogik.....	128
5.1.1	Schnittstelle für problemabhängige Implementierung	128
5.1.1.1	Probleminstanz-Daten	128
5.1.1.2	Lösung mit Subheuristiken	129
5.1.1.3	Globales Memory	132
5.1.2	Schnittstelle für problemunabhängige Implementierung	134
5.1.2.1	Hyperheuristik.....	134
5.1.2.2	Autoheuristik.....	135
5.1.3	Attribute.....	135
5.2	Slave-Programm.....	137
5.2.1	Grundlegende Funktionalität des Slave-Programms.....	137
5.2.2	Technische Realisierung der parallelen Prozesse	138
5.2.3	Verteilung der parallelen Prozesse im Framework	139
5.3	Master-Programm.....	140
5.3.1	Funktionalität der Benutzer-Oberfläche	140
5.3.1.1	Grundlegende Funktionen	141
5.3.1.2	Funktionen bezüglich Slaves	142
5.3.1.3	Funktionen zur Verwaltung von Lösungen und Slave-Tasks	143
5.3.1.4	Funktionen für einzelne manuelle Suchläufe	145
5.3.1.5	Funktionen bezüglich des Globalen Memory	145
5.3.1.6	Funktionen bezüglich der Historie von Lösungen.....	146
5.3.1.7	Funktionen für automatisierte Lösungsläufe.....	147
5.3.1.8	Funktionen zur Verwaltung von Parametern	149
5.3.2	Parallele Steuerung	150
5.3.2.1	Logische Steuerung	150
5.3.2.2	Physische Steuerung.....	152

5.3.3	Zusammenspiel mit dem Slave-Programm.....	153
6	IMPLEMENTIERUNG: SYSTEM UND NUTZUNG.....	157
6.1	Schnittstelle und Logik im Detail	157
6.1.1	Schnittstellenkomponente für die Daten.....	157
6.1.2	Schnittstellenkomponente für die Suchläufe	158
6.1.3	Schnittstellenkomponente für das Globale Memory.....	160
6.1.4	Schnittstellenkomponente für die Hyperheuristiken	161
6.1.5	Schnittstellenkomponente für die Autoheuristiken.....	164
6.1.6	Implementierte Autoheuristik	170
6.1.7	Nutzung der Autoheuristik-Klasse	174
6.2	Solver-Entwicklung mit pb.net.....	176
6.2.1	Evolutionäre Entwicklung.....	176
6.2.2	Kapselung der Subheuristiken.....	178
7	PRAXIS: EINSATZ DES FRAMEWORKS	181
7.1	Anwendung 1: Verteilte Entwicklung beim Pickup-and-Delivery-Problem....	181
7.1.1	Problemstellung	181
7.1.2	Entwicklung der Subheuristiken	183
7.1.3	Integration der Subheuristiken	184
7.1.4	Ergebnisse und Interpretation.....	188
7.2	Anwendung 2: Prototyping bei einem neuartigen Problem der Fahrer- und Fahrzeugeinsatzplanung	194
7.2.1	Problemstellung	194
7.2.2	Spezielle Metaheuristik als Subheuristik.....	199
7.2.3	(Re-)Konstruktionsheuristik.....	203
7.2.4	Integration und Implementierung von weiterer Lösungslogik	204
7.2.5	Spezielle Diversifikation.....	206
7.2.6	Erweiterung der Autoheuristik	207

7.2.7	Ausgestaltung der Testläufe und Ergebnisse	209
8	FAZIT: ZUSAMMENFASSUNG UND AUSBLICK.....	215
8.1	Anforderungen an die Zweckmäßigkeit des Frameworks	215
8.2	Grenzen der Funktionalität	216
8.3	Potentiale und Erweiterungsmöglichkeiten	217
9	LITERATURVERZEICHNIS	221
A	ANHANG: DEFINITION DER PB.NET-SCHNITTSTELLE.....	229
A.1	Komponenten der pb.net-Schnittstelle	229
A.2	Weiterführende Komponenten und Hilfsklassen der pb.net-Schnittstelle.....	230
A.3	Attribute der pb.net-Schnittstelle.....	232

II. Abbildungsverzeichnis

Abbildung 1: Globales und lokales Optimum	20
Abbildung 2: Crossover-Varianten	28
Abbildung 3: Aufbau des .NET-Frameworks.....	44
Abbildung 4: Gegenüberstellung von Walk- und Typen-Klassifikation	62
Abbildung 5: Die 3 Dimensionen der Taxonomie von Crainic, Toulouse und Gendreau ..	65
Abbildung 6: Verschiedene Kommunikationstopologien.....	67
Abbildung 7: Phasen der SWO	117
Abbildung 8: Realisierung eines verteilten nicht-hierarchischen Programmiermodells mit Master.....	127
Abbildung 9: Globales Memory mit zwei parallelen Suchprozessen.....	136
Abbildung 10: Benutzeroberfläche des Master-Programms	141
Abbildung 11: Zusammenspiel von Master- und Slave-Programm	151
Abbildung 12: Grundlegende Komponenten der pb.net-Schnittstelle.....	158
Abbildung 13: Weiterführende Komponenten und Hilfsklassen der pb.net-Schnittstelle	163

III. Tabellenverzeichnis

Tabelle 1: Testergebnisse für drei PDPTW-Instanzen.....	191
Tabelle 2: Testergebnisse für die zwölf Blockplanungsproblem-Instanzen	210

IV. Abkürzungsverzeichnis

1C	1-Control
ALNS	Adaptive Large Neighborhood Search
ASP	Application Server Pages
ALS	Abstract Local Search
BCL	Base Class Library
C	Collegial
CIL	Common Intermediate Language
CLR	Common Language Runtime
CLS	Common Language Specification
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
DS	Different Search
EA	Evolutionärer Algorithmus
FIFO	First-In-First-Out (-Prinzip)
GA	Genetischer Algorithmus
GIST	Greedy Indirect Search Technique
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identifikationsnummer
IDL	Interface Definition Language
KC	Knowledge Collegial
KS	Knowledge Synchronisation
ILS	Iterated Local Search
IP	Internet Protocol
JSP	Java Server Pages

LNS	Large Neighborhood Search
LP	Lineare Programmierung, Lineares Optimierungsproblem
MFC	Microsoft Foundation Classes
MP	Multiple Point
MSMQ	Microsoft Message Queuing
NP	Klasse der nichtdeterministisch lösbaren Probleme
pC	p-Control
PC	Personal Computer
PDPTW	Pickup and Delivery Problem with Time Windows
RCPSP	Resource Constraint Project Scheduling Problem
RAD	Rapid Application Development
RS	Rigid Synchronisation
SA	Simulated Annealing
SMP	Symmetrisches Multiprozessorsystem
SP	Single Point
SS	Same Search
SWO	Squeaky Wheel Optimization
TCP	Transmission Control Protocol
VND	Variable Neighborhood Descent
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows
XML	Extensible Markup Language