Matthias Rupp

Zeitoptimale Bearbeitungsreihenfolgen für mehrere Schweißroboter: Modelle und Algorithmen

Diplomarbeit, vorgelegt an der
Johann Wolfgang Goethe-Universität
Fachbereich Biologie und Informatik
Institut für Informatik
Professur für Syntaxanalyse und diskrete Strukturen

Version vom 5. Mai 2004.

Zusammenfassung

Das Thema der vorliegenden Diplomarbeit ist ein Problem aus der automatisierten Fertigung mit der Hilfe von Industrierobotern. Dabei sind n Aufgaben, in diesem Fall zu bearbeitende Schweißpunkte, von m Robotern so zu erledigen, dass die Gesamtbearbeitungszeit minimal ist. Die Roboter arbeiten gleichzeitig und dürfen einander nicht berühren. Das Problem enthält Aspekte aus den Gebieten Optimierung, Kollisionserkennung, Wegplanung und Robotersteuerung.

Die Schwerpunkte der Arbeit liegen auf der Modellierung des Problems, einem algorithmisch orientierten Überblick über die beteiligten Gebiete und einem empirischen Vergleich verschiedener Heuristiken zur Lösung eines stark vereinfachten kombinatorischen Modells. Das Problem wird zuerst exakt modelliert und anschließend systematisch vereinfacht. Der Überblick konzentriert sich auf die Gebiete Optimierung (neunzehn Verfahren), Kollisionserkennung (fünf Verfahren, neun Programmpakete) und Wegplanung (fünf Verfahren). Im Rahmen eines flexiblen evolutionären Ansatzes werden fünf stochastische Optimierungsverfahren in insgesamt dreizehn Varianten vorgestellt und auf elf Produktionsdatensätzen und achtzehn zufällig erzeugten Datensätzen empirisch analysiert. Die Arbeit enthält ein gegliedertes, kommentiertes Literaturverzeichnis.

Widmung

Meinen Eltern für ihre stete Unterstützung

Inhaltsverzeichnis

Inhaltsverzeichnis **vii**

Abbildungsverzeichnis **xiv**

Algorithmenverzeichnis **xv**

Tabellenverzeichnis **xvi**

Verwendete Symbole **xviii**

Vorwort **xxi**

1 Das Schweißzellenproblem **1**

2 Klassifikation und Ansätze **15**

3 Modellierung **33**

4 Ein evolutionärer Ansatz **47**

5 Parametersätze und Analyse **83**

6 Ausblick & Fazit **121**

A Statistische Kenngrößen **123**

Literaturverzeichnis **139**

Inhaltsverzeichnis

Inhaltsverzeichnis **vii**

Abbildungsverzeichnis **xiv**

Algorithmenverzeichnis **xv**

Tabellenverzeichnis **xvi**

Verwendete Symbole **xviii**

Vorwort **xxi**
Zielsetzung . . . xxi
Danksagung . . . xxi
Formalia . . . xxi

1 Das Schweißzellenproblem **1**
1.1 Hintergrund und Beschreibung . . . 1
1.2 Erweiterungen . . . 3
1.2.1 Sicherheitsabstand . . . 3
1.2.2 Hindernisse . . . 3
1.2.3 Schweißzeiten . . . 3
1.2.4 Erweiterte Problembeschreibung . . . 4
1.3 Schwierigkeiten . . . 4
1.3.1 Gegenseitige Beeinflussung der Roboter . . . 4
1.3.2 Unterschiedliche Wege und Wegzeiten . . . 4
1.3.3 Wegplanung und Ansteuerung . . . 5
1.3.4 Modellierung der Roboter . . . 5
1.4 Formalisierung . . . 5
1.4.1 Konfigurationen der Roboter . . . 5
1.4.2 Roboter und Schweißpunkte . . . 6
1.4.3 Koordinatensysteme und Abstände . . . 7
1.4.4 Nebenbedingungen . . . 7
1.4.5 Kontinuierliche Formulierung . . . 8
1.4.6 Diskrete Formulierung . . . 9
1.4.7 Vergleich und Anmerkungen . . . 11
1.5 Weiterführendes . . . 12
Mehrere Robotertypen . . . 12
Werkzeugwechsel . . . 12
Fliegender Wechsel . . . 12
Feste Schweißvorrichtung . . . 12

Laserschweißen . . . 13
Andere Arbeitsaufgaben . . . 13
Weitere Optimierungskriterien . . . 13
Verschiedene Lösungen . . . 13
Stationsplanung . . . 13

2 Klassifikation und Ansätze 15
2.1 Kollisionserkennung . . . 15
2.1.1 Gegenstand des Gebiets . . . 15
2.1.2 Einordnung des Schweißzellenproblems . . . 16
2.1.3 Methoden . . . 16
Hüllvolumen-Bäume . . . 16
Lin-Canny-Algorithmus . . . 17
Voronoi-Clip-Algorithmus . . . 17
Gjk-Algorithmus . . . 17
Aufteilung des Raums . . . 17
2.1.4 Implementierungen . . . 18
FreeSolid . . . 18
Opcode . . . 18
QuickCd . . . 18
Rapid . . . 18
PQP . . . 19
Swift++ . . . 19
V-Clip . . . 19
V-Collide . . . 19
Verbesserter GJK . . . 19
2.2 Wegplanung und Ansteuerung . . . 19
2.2.1 Gegenstand des Gebiets . . . 20
2.2.2 Einordnung des Schweißzellenproblems . . . 20
2.2.3 Komplexität . . . 20
2.2.4 Methoden . . . 21
A^*-Algorithmus . . . 21
Zell-Dekomposition . . . 21
Stochastische Straßenkarte . . . 21
Potentialfeld . . . 22
Randomisierte Wegplaner . . . 22
2.3 Optimierung . . . 22
2.3.1 Gegenstand des Gebiets . . . 23
2.3.2 Einordnung des Schweißzellenproblems . . . 23
2.3.3 Methoden zur globalen Optimierung . . . 24
2.3.3.1 Deterministische Methoden . . . 24
Dynamisches Programmieren . . . 24
Branch & Bound . . . 24
Lineare Programmierung . . . 25
Nichtlineare Programmierung . . . 25
Intervall-Methoden . . . 25
Fortsetzungs- und Glättungsmethoden . . . 25
2.3.3.2 Stochastische Methoden . . . 26
Multistart und Clustering . . . 26
Kontrollierte Zufällige Suche . . . 26
Metropolis-Algorithmus . . . 26

Simuliertes Abkühlen . . . 26
Baum Abkühlen . . . 26
Evolutionäre Algorithmen . . . 27
Scatter Search . . . 27
Ameisen-Algorithmen . . . 27
Statistische Methoden . . . 27
2.3.3.3 Hybride Verfahren und Metaheuristiken . . . 28
Tabu-Suche . . . 28
Memetische Algorithmen . . . 28
Iterierte lokale Suche . . . 28
Grasp . . . 28
2.3.4 Verwandte Probleme . . . 28
2.3.4.1 Das Problem des Handlungsreisenden . . . 28
2.3.4.2 Routenplanungsprobleme für Fahrzeuge . . . 30
2.3.4.3 Ablaufsteuerung und -planung . . . 31
2.4 Weiterführendes . . . 31

3 Modellierung **33**
3.1 Vereinfachungsmöglichkeiten . . . 33
3.1.1 Optimalität . . . 33
3.1.2 Kollisionsfreiheit . . . 34
3.1.3 Nebenbedingungen . . . 34
3.1.4 Sicherheitsabstand . . . 34
3.1.5 Hindernisse . . . 34
3.1.6 Schweißzeiten . . . 35
3.1.7 Gegenseitige Beeinflussung der Roboter . . . 35
3.1.8 Unterschiedliche Wege und Wegzeiten . . . 35
3.1.9 Wegplanung und Ansteuerung . . . 36
3.1.10 Modellierung der Roboter . . . 36
3.2 Anforderungen an Algorithmen . . . 36
3.2.1 Maschinenmodell . . . 37
3.2.2 Analyse . . . 37
3.3 Vereinfachte Problemvarianten . . . 37
3.3.1 Variante WCP-A . . . 37
3.3.1.1 Definition . . . 38
3.3.1.2 Komplexität . . . 39
3.3.2 Variante WCP-B . . . 42
3.3.3 Variante WCP-C . . . 44
3.4 Weiterführendes . . . 45

4 Ein evolutionärer Ansatz **47**
4.1 Wahl des Optimierungsverfahrens . . . 47
4.2 Grundbegriffe der evolutionären Optimierung . . . 49
4.2.1 Genotyp, Phänotyp, Individuum . . . 49
4.2.2 Population, Generation . . . 50
4.2.3 Selektion, Fitness . . . 50
4.2.4 Rekombination, Mutation . . . 50
4.3 Anmerkungen . . . 50
4.3.1 Kodierung . . . 51
4.3.2 Neutrale Evolution . . . 51
4.3.3 Diversität . . . 51

4.3.4 Nebenbedingungen . . . 52
4.4 Bausteine für einen evolutionären Ansatz . . . 53
4.4.1 Ablauf . . . 53
4.4.2 Kombination mehrerer Algorithmen . . . 54
4.4.3 Kodierung . . . 55
4.4.4 Populationsgröße und Anzahl der Nachkommen . . . 56
4.4.5 Initialisierung der Population . . . 57
4.4.5.1 Uniforme Verteilung . . . 57
4.4.5.2 Nächster Roboter & Gierige Ersparnis-Heuristik . . . 58
4.4.6 Fitnessfunktion . . . 61
4.4.7 Terminierungsbedingung . . . 62
Maximale Anzahl an Generationen . . . 62
Keine Verbesserung der besten Individuen . . . 62
Maximale Anzahl an Auswertungen der Fitnessfunktion . . . 62
4.4.8 Selektion . . . 63
4.4.8.1 Paarungsselektion . . . 63
4.4.8.2 Umweltselektion . . . 64
4.4.8.3 Selektionsmethoden . . . 64
Deterministische Selektion des besten Individuums . . . 64
Uniforme Selektion . . . 64
Rangbasierte Selektion . . . 64
Lineare rangbasierte Selektion . . . 65
Exponentielle rangbasierte Selektion . . . 65
Stochastische Turnierselektion . . . 67
Andere Selektionsmethoden . . . 67
4.4.9 Rekombination . . . 67
4.4.9.1 Identität . . . 68
4.4.9.2 Kantenrekombination . . . 68
4.4.10 Mutation . . . 72
4.4.10.1 Identität . . . 72
4.4.10.2 Inversion . . . 72
4.4.10.3 Transposition . . . 74
Transposition zwischen Partitionen . . . 74
Transposition innerhalb einer Partition . . . 74
4.4.10.4 Populationsgesteuerte Inversion und Transposition . . . 76
4.4.10.5 Simuliertes Abkühlen . . . 78
4.4.10.6 Iterierte lokale Suche . . . 79
4.5 Weiterführendes . . . 80

5 Parametersätze und Analyse **83**
5.1 Parameter und Abhängigkeiten . . . 83
5.2 Datensätze . . . 86
5.3 Auswertung . . . 87
5.3.1 Kriterien . . . 87
5.3.2 Evaluation eines Algorithmus . . . 87
5.3.2.1 Grafische Darstellung . . . 88
5.3.3 Vergleich mehrerer Algorithmen . . . 88
5.3.3.1 Grafische Darstellung . . . 88
5.3.3.2 Direkter Vergleich . . . 89
5.3.3.3 Gegenüberstellung der Funktionsauswertungen . . . 89
5.3.4 Interpretation . . . 90

5.4 Parametersätze . . . 90
5.4.1 Evolutionäre Algorithmen . . . 90
5.4.2 Populationsbasierte Ansätze . . . 94
5.4.3 Simuliertes Abkühlen . . . 100
5.4.4 Iterierte lokale Suche . . . 106
5.4.4.1 Eine stochastische Variante . . . 107
5.4.4.2 Iterierte lokale Optimierung . . . 110
5.4.5 Memetische Algorithmen . . . 112
5.5 Gesamtauswertung . . . 113
5.5.1 Auszuwertendes Datenmaterial . . . 113
5.5.2 Untere Schranken und Skalierung der Ergebnisse . . . 113
5.5.3 Rangbasierte Bewertung der Algorithmen . . . 113
5.6 Ausblick . . . 118

6 Ausblick & Fazit **121**
6.1 Zusammenfassung . . . 121
6.2 Ausblick . . . 121
6.3 Fazit . . . 122

A Statistische Kenngrößen **123**
A.1 Schätzer für Erwartungswert und Varianz . . . 123
A.2 Konfidenzintervalle . . . 124

Literaturverzeichnis **139**
Kollisionserkennung . . . 139
Überblick & Allgemeines . . . 139
Hüllvolumen-Ansätze . . . 139
GJK-Algorithmus . . . 141
Lin-Canny-Algorithmus . . . 142
Wegplanung . . . 142
Überblick und Allgemeines . . . 142
Stochastische Straßenkarten . . . 143
Andere Methoden . . . 143
Robotik . . . 144
Bahnplanung . . . 144
Roboter . . . 144
Optimierung . . . 144
Überblick und Allgemeines . . . 144
Verzweige & Begrenze-Algorithmen . . . 145
Mathematische Programmierung . . . 146
Fortsetzungs- und Glättungsmethoden . . . 146
Simuliertes Abkühlen . . . 147
Evolutionäre Algorithmen . . . 147
Tabu-Suche . . . 149
Memetische Algorithmen . . . 150
Iterierte lokale Suche . . . 150
Grasp . . . 150
Andere Methoden . . . 151
TSP . . . 152
Andere Optimierungsprobleme . . . 153
Verschiedenes . . . 153

Grundlagen 153
Komplexitätstheorie 154
Weitere Quellen 155

Abbildungsverzeichnis

1.1 Zusammensetzung des Weges von M_i in der diskreten Formulierung des WCP . . . 9

2.1 Repräsentationsformen geometrischer Objekte 16

3.1 Zusammensetzung des Weges von M_i in Variante WCP-B des WCP 42

4.1 Ablaufschema evolutionärer Algorithmen 49
4.2 Beispiel für die nächster Roboter & gierige Ersparnis-Heuristik 60
(a) Station 60
(b) Kantengewichte 60
4.3 Lineare rangbasierte Selektion 66
4.4 Exponentielle rangbasierte Selektion 66
4.5 Stochastische Turnierselektion 68
4.6 Äquivalenz von Inversionen 73
4.7 Trennstellen für die Inversion 73
4.8 Akzeptanzwahrscheinlichkeiten beim simulierten Abkühlen 80

5.1 Berechnungsverlauf von Algorithmus A_1 auf den Datensätzen E und f 92
5.2 Berechnungsverläufe der Algorithmen A_1 und A_2 auf den Datensätzen E und f . . 93
5.3 Berechnungsverläufe der Algorithmen A_2 und B_1 auf dem Datensatz E 96
5.4 Berechnungsverläufe der Algorithmen C_1, C_2 und C_3 auf den Datensätzen A und f 103
5.5 Berechnungsverläufe der Algorithmen C_5 und D_1 auf den Datensätzen E und e . . 108

Algorithmenverzeichnis

4.1 Ein evolutionärer Algorithmus für das WCP-A 54
4.2 Ressourcenverteilung bei absoluten und relativen Anforderungen 54
4.3 Uniforme Erzeugung von Lösungskandidaten 57
4.4 Nächster Roboter & gierige Ersparnis-Heuristik 59
4.5 Paarungsselektion für mehrere Selektionsmethoden 64
4.6 Zwei Varianten der Turnierselektion 67
4.7 Kantenrekombination für das WCP-A 70
4.8 Inversion einer angeordneten Partition eines Individuums 74
4.9 Transposition zwischen zwei Partitionen 75
4.10 Tao/Michalewicz-Heuristik für das TSP 76
4.11 Populationsgesteuerte Inversion und Transposition 77
4.12 Simuliertes Abkühlen 79
4.13 Iterierte lokale Suche 81
5.1 Erstellung von Ranglisten 115

5.1 Algorithmus A_1 (evolutionärer Algorithmus, Variante 1) 91
5.2 Algorithmus A_2 (evolutionärer Algorithmus, Variante 2) 91
5.3 Algorithmus B_1 (populationsbasierter Ansatz, Variante 1) 95
5.4 Algorithmus B_2 (populationsbasierter Ansatz, Variante 2) 97
5.5 Algorithmus B_3 (populationsbasierter Ansatz, Variante 3) 99
5.6 Algorithmus C_1 (simuliertes Abkühlen, Variante 1) 100
5.7 Algorithmus C_2 (simuliertes Abkühlen, Variante 2) 100
5.8 Algorithmus C_3 (simuliertes Abkühlen, Variante 3) 101
5.9 Algorithmus C_4 (simuliertes Abkühlen, Variante 4) 104
5.10 Algorithmus C_5 (simuliertes Abkühlen, Variante 5) 105
5.11 Algorithmus D_1 (iterierte lokale Suche, Variante 1) 107
5.12 Algorithmus D_2 (iterierte lokale Suche, Variante 2) 109
5.13 Algorithmus E_1 (memetischer Algorithmus, Variante 1) 111

Tabellenverzeichnis

1 Übersicht der verwendeten Symbole xviii

1.1 Zulässige Winkelstellungen des KR 150 K 6

2.1 Programmbibliotheken zur Kollisionserkennung 18

4.1 Eignung von Optimierungsverfahren für das WCP-A 48
4.2 Einschränkungen bei Populationsgröße und Anzahl der Nachkommen 56
4.3 Unterschiede zwischen Paarungsselektion und Umweltselektion 63
4.4 Wahrscheinlichkeiten der Knotenzuordnungen in Beispiel 4.4 69
4.5 Erstellung der Kantentabelle in Beispiel 4.4 71
(a) Multimengen Q_j der Knoten 71
(b) Multimengen R_i der Startknoten 71

5.1 Angabe von Parametersätzen 84
5.2 Datensätze für das WCP-A 86
(a) Produktionsdaten 86
(b) Zufällige Datensätze 86
5.3 Direkter Vergleich der Ergebnisgüte von Algorithmus A_1 und Algorithmus A_2 . . . 94
5.4 Direkter Vergleich der Ergebnisgüte von Algorithmus A_2 und Algorithmus B_1 . . . 96
5.5 Direkter Vergleich der Ergebnisgüte von Algorithmus A_2 und Algorithmus B_2 . . . 97
5.6 Direkter Vergleich der Ergebnisgüte von Algorithmus A_2 und Algorithmus B_3 . . . 98
5.7 Anzahl der Funktionsauswertungen von Algorithmus A_2 und Algorithmus B_3 . . . 99
5.8 Direkter Vergleich der Ergebnisgüte von Algorithmus C_1 und Algorithmus C_3 . . . 101
5.9 Direkter Vergleich der Ergebnisgüte von Algorithmus C_3 und Algorithmus C_4 . . . 102
5.10 Anzahl der Funktionsauswertungen von Algorithmus C_3 und Algorithmus C_4 . . . 104
5.11 Direkter Vergleich der Ergebnisgüte von Algorithmus C_4 und Algorithmus C_5 . . . 105
5.12 Anzahl der Funktionsauswertungen von Algorithmus C_4 und Algorithmus C_5 . . . 106
5.13 Direkter Vergleich der Ergebnisgüte von Algorithmus C_5 und Algorithmus D_1 . . . 109
5.14 Direkter Vergleich der Ergebnisgüte von Algorithmus D_1 und Algorithmus D_2 . . . 110
5.15 Direkter Vergleich der Ergebnisgüte von Algorithmus A_2 und Algorithmus E_1 . . . 111
5.16 Direkter Vergleich der Ergebnisgüte von Algorithmus D_2 und Algorithmus E_1 . . . 112
5.17 Obere und untere Schranken 114
5.18 Rangliste aller Algorithmen 116
5.19 Anzahl der Funktionsauswertungen von Algorithmus C_5 und Algorithmus D_1 . . . 117
5.20 Anzahl der Funktionsauswertungen von Algorithmus D_1 und Algorithmus D_2 . . . 118

A.1 Statistische Kenngrößen für Algorithmus A_1 125
A.2 Statistische Kenngrößen für Algorithmus A_2 126
A.3 Statistische Kenngrößen für Algorithmus B_1 127
A.4 Statistische Kenngrößen für Algorithmus B_2 128

A.5 Statistische Kenngrößen für Algorithmus B_3 129
A.6 Statistische Kenngrößen für Algorithmus C_1 130
A.7 Statistische Kenngrößen für Algorithmus C_2 131
A.8 Statistische Kenngrößen für Algorithmus C_3 132
A.9 Statistische Kenngrößen für Algorithmus C_4 133
A.10 Statistische Kenngrößen für Algorithmus C_5 134
A.11 Statistische Kenngrößen für Algorithmus D_1 135
A.12 Statistische Kenngrößen für Algorithmus D_2 136
A.13 Statistische Kenngrößen für Algorithmus E_1 137

Verwendete Symbole

Symbole, die in dieser Arbeit definiert werden, sind mit der Seitenangabe ihrer Definition versehen. Geläufige mathematische Symbole, die in der Arbeit benutzt, aber nicht definiert werden, sind ohne Seitenangabe gelistet.

Symbol	Bedeutung	Seite
$\mathbb{N}$	Menge der natürlichen Zahlen ohne die Null	
$\mathbb{N}_0$	Menge der natürlichen Zahlen einschließlich der Null	
$\mathbb{R}$	Menge der reellen Zahlen	
$\mathbb{R}_{\geq 0}$	Menge der nicht negativen reellen Zahlen	
$\mathbb{R}_{>0}$	Menge der positiven reellen Zahlen	
$[a, b]$	Abgeschlossenes Intervall (einschliesslich der Intervallgrenzen a und b)	
$]a, b[$	Offenes Intervall (ausschliesslich der Intervallgrenzen a und b)	
$]a, b]$	Halboffenes Intervall (ausschließlich a, einschließlich b)	
$[a, b[$	Halboffenes Intervall (einschließlich a, ausschließlich b)	
$[\,]$	Gaußklammer (rundet kaufmännisch)	
$\lfloor\,\rfloor$	Untere Gaußklammer (rundet ab)	
$\lceil\,\rceil$	Obere Gaußklammer (rundet auf)	
$\binom{n}{k}$	Binomialkoeffizient. Anzahl der Möglichkeiten für die Wahl von k aus n Elementen ohne Wiederholung und ohne Reihenfolge	
$\left(\!\binom{n}{k}\!\right)$	Multinomialkoeffizient. Anzahl der Möglichkeiten für die Wahl von k aus n Elementen mit Wiederholung und ohne Reihenfolge	
$\mathbb{E}[X]$	Erwartungswert der Zufallsvariablen X	
$\mathbb{V}[X]$	Varianz der Zufallsvariablen X	
$\mathbb{P}[A]$	Wahrscheinlichkeit des Ereignisses A	
P	Komplexitätsklasse der polynomiell zeitbeschränkten Programme für deterministische Turingmaschinen	
NP	Komplexitätsklasse der polynomiell zeitbeschränkten Programme für nicht deterministische Turingmaschinen	
ϵ	Das leere Wort	
Σ^*	Kleenescher Abschluss (Menge aller Worte über Σ, einschließlich ϵ)	

Fortsetzung auf der nächsten Seite...

Tabelle 1: Übersicht der verwendeten Symbole

Symbol	Bedeutung	Seite
a_j	Menge aller Roboter, die J_j bearbeiten dürfen (engl. allowed)	7
$B_i(j)$	Anfangszeit der Bearbeitung von $J_{\sigma_i(j)}$ durch M_i in der diskreten Formulierung (engl. begin of processing)	10
$\mathcal{C}$	Konfigurationsraum eines Roboters (engl. configuration space)	5
C_E	Menge der Ausschluss-Nebenbedingungen (engl. exclusion constraints)	7
C_S	Menge der Reihenfolge-Nebenbedingungen (engl. sequence constraints)	7
d_i	Abstand von Roboter M_i zum Werkstück (engl. distance)	7
$d_{i,j}$	Abstand von Roboter M_i zum Roboter M_j (engl. distance)	7
$E_i(j)$	Endzeit der Bearbeitung von $J_{\sigma_i(j)}$ durch M_i in der diskreten Formulierung (engl. end of processing)	10
f_i	Bewegung von M_i in der kontinuierlichen bzw. diskreten Formulierung	8,10
$f_{i,j}$	Zwischenbewegung von M_i in der diskreten Formulierung	10
$\mathcal{G}$	Menge aller Objektgeometrien (engl. geomtry)	6
J	Menge aller Schweißpunkte (engl. job)	6
J_i	Der i-te Schweißpunkt der Station (engl. job)	6
κ	Anzahl der Gelenke eines Roboters; Dimension des Konfigurationsraums	5
L_i	Die M_i zugeordneten Schweißpunkte in der diskreten Formulierung	10
$L(j)$	Partition von J_j in der diskreten Formulierung	10
$\mathcal{M}$	Menge aller ansteuerbaren Trajektorien (engl. movement)	8
m	Anzahl der Roboter (engl. machines)	6
M_i	Der i-te Roboter der Station (engl. machine)	6
n	Anzahl der Schweißpunkte (engl. number of welding points).	6
σ_i	Anordnung von L_i in der diskreten Formulierung (engl. sequence)	10
σ_i^{-1}	Umkehrabbildung von σ_i in der diskreten Formulierung	10
s	Sicherheitsabstand (engl. safety distance)	3
S_i	Anfangsstellung (auch Startstellung) von M_i (engl. starting stance)	6
s_j	Schweißzeitpunkt von J_j in der kontinuierlichen Formulierung	8
t_j	Schweißzeit für J_j (engl. welding time)	6
$t_{\max}$	Gesamtbearbeitungszeit der Lösung (engl. maximum time)	8,10
$\mathcal{V}_{i,j}$	Die Menge aller geeigneten Stellungen von M_i auf J_j (engl. valid stances)	6
$v_{i,j}$	Schweißstellung von M_i auf $J_{\sigma_i(j)}$ in der diskreten Formulierung	10
$w_{i,j}$	Wegzeit für $f_{i,j}$ in der diskreten Formulierung (engl. way time)	10
f	Zu optimierende Funktion innerhalb eines Optimierungsproblems	23
$\mathcal{S}$	Suchraum eines Optimierungsproblems (engl. search space)	23
$\mathcal{F}$	Menge aller Individuen	55
f	Fitnessfunktion (engl. fitness function)	53,61
$f_{\text{best}}^{(i)}$	Bester Fitnesswert bis Generation i einschließlich (engl. best fitness)	53
$f_{\text{eval}}^{(i)}$	Anzahl der (verzögerten) Auswertungen der Fitnessfunktion in Generation i (engl. fitness evaluations)	61
$f_{\max}$	Maximale Anzahl an Auswertungen der Fitnessfunktion	62
$f_{\max}^{(i)}$	Bester Fitnesswert in Generation i (engl. maximum fitness)	53
I_k	Das k-te Individuum einer Population (engl. individual)	53
λ	Anzahl der Nachkommen in einer Generation	53

Fortsetzung auf der nächsten Seite...

Tabelle 1: Übersicht der verwendeten Symbole

Symbol	Bedeutung	Seite
λ'	Faktor zur Bestimmung von λ	56
μ	Größe der Population	53
μ'	Faktor zur Bestimmung von μ	56
$\tau - 1$	Anzahl bereits simulierter Generationen	53
τ_{last}	Maximale Anzahl Generationen ohne Verbesserung des besten Individuums	62
τ_{max}	Maximale Anzahl simulierter Generationen	62
$\mathfrak{f}^{(k)}$	Vektor der Ergebnisse in Generation k aller analysierten Programmläufe	88
$\mathfrak{f}^{(k)}_{\text{max}}$	Schlechtestes Ergebnis für Generation k in allen analysierten Programmläufen	88
$\mathfrak{f}^{(k)}_{\text{min}}$	Bestes Ergebnis für Generation k in allen analysierten Programmläufen	88
$\hat{\mu}$	Schätzer für den Erwartungswert	123
$\hat{\sigma}^2$	Schätzer für die Varianz	123
min_{d}	Bestes Ergebnis auf Datensatz d	113

TABELLE 1: Übersicht der verwendeten Symbole.

Vorwort

Zielsetzung

Die vorliegende Diplomarbeit behandelt ein industriell relevantes Problem unter theoretischen und praktischen Gesichtspunkten. Sie verfolgt dabei mehrere Zielsetzungen: Das Problem wird dem Kenntnisstand des Autors nach in dieser Form so erstmalig untersucht; daher liegt ein Schwerpunkt auf der Formalisierung (Kapitel 1) und Modellierung (Kapitel 3) der Problemstellung sowie der Entwicklung (Kapitel 3) und empirischen Untersuchung (Kapitel 5) geeigneter Verfahren zu seiner Lösung. Weiterhin soll die Arbeit als Ausgangspunkt für weitergehende Untersuchungen des Themas dienen können. Sie enthält zu diesem Zweck einen ausführlicheren Überblick über Verfahren und Problemstellungen beteiligter Wissensgebiete (Kapitel 2) sowie eine Auflistung offener Probleme und weiterführender Fragestellungen (Kapitel 6).

Danksagung

Ich danke all den Menschen, deren Unterstützung diese Arbeit möglich gemacht hat, namentlich

- Meinen Eltern, deren Unterstützung mir das Studium erst ermöglicht hat.
- Dipl.-Math. Bernhard Mauersberg und Dr. Wolfgang Mergenthaler von der Firma *Beratende Ingenieure Frankfurt* für die gute Zusammenarbeit sowie Prof. Dr. Anton Wakolbinger für die Vermittlung.
- Den nachfolgenden Personen (in alphabetischer Reihenfolge der Nachnamen) für Diskussionen, Anregungen, Ratschläge, Ideen, Literaturhinweise und Korrekturen

Simone Galluba	Antonin Nemec
Prof. Dr. Rainer Kemp	Ingo Ritter
Prof. Dr. Detlef Krömker	Dipl.-Math. Thomas Rupp
Uli Laube	Prof. Dr. Georg Schnitger
Prof. Dr.-Ing. Frank Mantwill	Dipl.-Ing. (FH) Rico Tunk
PD Dr. Markus Nebel	Björn Weber

sowie bei allen, deren Erwähnung ich vergessen haben sollte.

Formalia

Die Arbeit ist, obwohl ich sie als einzelne Person verfasst habe, der leichteren Lesbarkeit willen wie im wissenschaftlichen Bereich üblich in der 1. Person Plural geschrieben. Gegebenenfalls möge man den Plural einfach als „der Autor und der geneigte Leser“ interpretieren.

Definitionen, Sätze, Lemmata, Beweise und Beispiele sind durch ein Kästchen □ als optische Begrenzung abgeschlossen. Die Arbeit wurde mit dem Textsatzprogramm LaTeX gesetzt.

Matthias Rupp, 6. Mai 2004

Kapitel 1

Das Schweißzellenproblem

Wir beginnen mit einer informellen Beschreibung der untersuchten Problemstellung und geben anschließend eine möglichst exakte formale Definition. Diese wird in späteren Kapiteln vereinfacht, um damit arbeiten zu können. Zweck der formalen Definition ist es, das Problem präzise zu erfassen und den Vergleich mit späteren, vereinfachten Modellen zu erlauben und somit deren Beurteilung zu erleichtern. Zuletzt wird auf Variationsmöglichkeiten des Problems eingegangen.

1.1 Hintergrund und Beschreibung

In der Automobilbranche werden bei der automatisierten Fertigung von Fahrzeugen Industrieroboter[1] eingesetzt. Diese sind entlang von Produktionsstraßen zu Arbeitsgruppen, sogenannten *Stationen*, zusammengefasst. Jeder Station wird eine Menge von Arbeitsschritten an einem die Produktionsstraße durchlaufenden Werkstück zugewiesen; diese werden von den Robotern der Station parallel bearbeitet.

In dieser Arbeit geht es um eine spezielle Ausprägung dieses Szenarios, das automatisierte Punktschweißen. Dabei sind die Arbeitsschritte Schweißpunkte, d. h. ausgezeichnete Stellen eines metallenen Werkstücks, die von einem Roboter angefahren und mit Hilfe eines speziellen Werkzeugs, der Schweißzange, dauerhaft verbunden werden.

Bei den Robotern handelt es sich um Knickarmroboter mit typischerweise fünf oder sechs Gelenken, die jeweils einen Freiheitsgrad haben. Die Roboter selbst sind baugleich, während die Schweißzangen unterschiedlich sein können (und dann für unterschiedliche Arten von Schweißpunkten zuständig sind). Sie arbeiten nach dem Prinzip des elektrischen Widerstandsschweißens; dabei werden die überlappenden Metallteile des Werkstücks von den beiden stromführenden Elektrodenspitzen an den Enden der Schweißzange eingeklemmt und verbinden sich an dieser Stelle aufgrund des ausgeübten Druckes und der durch den eigenen elektrischen Widerstand erzeugten Wärme. [87]

Das betrachtete Problem besteht darin, die vorgegebenen Schweißpunkte so durch die Roboter einer Station bearbeiten zu lassen, dass die Gesamtbearbeitungszeit unter gewissen Nebenbedingungen minimiert wird; letztere ergeben sich hauptsächlich aus technischen Beschränkungen. Insbesondere muss die Aufgabenverteilung physikalisch durch die Station realisierbar sein: Roboter dürfen nicht miteinander oder mit dem Bauteil kollidieren, sie müssen die Ausführungsorte ihrer Arbeitsschritte erreichen können etc. Weitere Nebenbedingungen resultieren aus produktionstechnischen Anforderungen, z. B. müssen manche Schweißpunkte vor anderen bearbeitet werden.

[1] Industrieroboter sind nach der VDI-Richtlinie 2860 „universell einsetzbare Bewegungsautomaten mit mehreren Achsen, deren Bewegungen hinsichtlich Bewegungsfolge und Wegen bzw. Winkeln frei (d. h. ohne mechanischen Eingriff) programmierbar und gegebenenfalls sensorgeführt sind. Sie sind mit Greifern, Werkzeugen oder anderen Fertigungsmitteln ausrüstbar und können Handhabungs- und/oder Fertigungsaufgaben ausführen.“

Eine Station wird durch die Anordnung[2] ihrer Roboter sowie deren Geometrie und Kenndaten beschrieben. Die Stellung eines Roboters im Raum ist durch die Winkelstellungen seiner Gelenke eindeutig bestimmt. Zu Beginn befinden sich die Roboter in vorgegebenen Ausgangsstellungen. Die Position des Bauteils ist für die Dauer der Bearbeitung fest. Das Schweißen eines Schweißpunktes darf nicht unterbrochen werden.

Bevor wir uns eingehender mit den einzelnen Punkten beschäftigen, fassen wir sie kurz in einer ersten Definition zusammen:

Definition 1.1 (Problembeschreibung) Gegeben eine Station, beschrieben durch Anzahl, Positionen, Ausrichtungen, Geometrie und Kenndaten ihrer Roboter (baugleichen Typs), eine Menge von Schweißpunkten sowie eine Menge von Nebenbedingungen. Finde eine Ansteuerung der Roboter, welche die folgenden Bedingungen erfüllt:

(1) *Optimalität:* Die Gesamtbearbeitungszeit, d. h. die Zeit, die vergeht, bis alle Roboter ihre Arbeit beendet haben, muss minimal sein.

(2) *Vollständigkeit:* Am Ende müssen alle Schweißpunkte bearbeitet worden sein.

(3) *Kollisionsfreiheit:* Kein Roboter darf mit sich selbst, mit einem anderen Roboter oder mit dem Bauteil kollidieren.

(4) *Nebenbedingungen:* Die folgenden Arten von Nebenbedingungen sind zulässig:
   (a) *Reihenfolge:* Ein Schweißpunkt muss vor einem anderen bearbeitet werden.
   (b) *Ausschluss:* Ein Roboter darf einen Schweißpunkt nicht bearbeiten.

□

Im Automobilbau hat eine Station typischerweise zwischen zwei und sechs, höchstens zwölf Roboter und ein Werkstück 20 bis 500 Schweißpunkte. Die verwendeten Roboter haben meist fünf oder sechs Gelenke. Ein häufig eingesetztes Modell ist der sechsachsige Industrieroboter KR 150 der Firma *KUKA Roboter GmbH*. [30, 31]

Das Problem enthält mehrere (miteinander zusammenhängende) Teilaufgaben:

(a) Verteilen der Schweißpunkte auf die Roboter.

(b) Festlegen von Bearbeitungsreihenfolgen für die Schweißpunkte eines jeden Roboters.

(c) Erkennen von Kollisionen.

(d) Planen von Wegen für die Roboter (Umgehen von Hindernissen, Koordination mehrerer Roboter, Verhindern von Kollisionen).

(e) Programmierung (Ansteuerung) der Roboter.

Der Schwerpunkt dieser Arbeit liegt auf den Punkten a und b, die wir zusammen als kombinatorisches Optimierungsproblem auffassen. Die Punkte c, d und e bezeichnen wir als die Probleme der Kollisionserkennung, Wegplanung und Ansteuerung.

[2] Dazu gehören Position und Ausrichtung eines Roboters. Die Position bezieht sich auf einen definierten Punkt des Roboters, den Fußpunkt. Die aus drei Winkelangaben bestehende Ausrichtung legt fest um wieviel Grad das Koordinatensystem des Roboters um die drei Achsen gedreht ist. Es gibt unterschiedliche Konventionen für die Reihenfolge von Drehungen und Translation sowie für die Wahl der Rotationsachsen.

1.2 Erweiterungen

Im Folgenden erweitern wir das Problem um einige naheliegende Aspekte.

1.2.1 Sicherheitsabstand

Wird der Abstand, den die Roboter während der Ausführung voneinander und vom Werkstück haben, sehr klein, kann es in der Praxis bereits zu Kollisionen kommen, z. B. aufgrund mechanischer Verformungen, Abweichungen bei der Ausführung des Programms oder Umwelteinflüssen. Es empfiehlt sich daher, einen Sicherheitsabstand einzuhalten.

Diese Forderung ist gut handhabbar und lässt sich als Parameter in die Kollisionserkennung integrieren. Wir modifizieren dazu die Problembeschreibung wie folgt:

Definition 1.2 (Erweiterung um Sicherheitsabstand) In der Problembeschreibung in Definition 1.1 ist zusätzlich ein *Sicherheitsabstand* $s \in \mathbb{R}_{\geq 0}$ gegeben. Punkt 3 wird ersetzt durch

(3) *Kollisionsfreiheit.* Kein Roboter darf mit sich selbst kollidieren. Zu keinem Zeitpunkt darf ein Roboter zu einem anderen Roboter oder zu dem Bauteil einen Abstand kleiner oder gleich s haben.

□

Die ursprüngliche Formulierung ergibt sich als Spezialfall mit $s = 0$.

1.2.2 Hindernisse

Im Produktionsbetrieb sind die Roboterstationen in die Fertigungsstraßen von Werkshallen eingebunden. Diese Umgebung kann die Bewegungsmöglichkeiten der Roboter durch Hindernisse wie Teile der Hallenarchitektur (Träger, Säulen, Sockel, Wände usw.) oder andere Maschinen einschränken.

Hindernisse lassen sich im Rahmen der Kollisionserkennung und Wegplanung handhaben. Wir formulieren die Modifikation der Problembeschreibung zunächst unabhängig von der Erweiterung um den Sicherheitsabstand:

Definition 1.3 (Erweiterung um Hindernisse) In der Problembeschreibung in Definition 1.1 ist zusätzlich die Geometrie der Station gegeben. Punkt 3 wird ersetzt durch

(3) *Kollisionsfreiheit.* Kein Roboter darf mit sich selbst, mit einem anderen Roboter, mit dem Bauteil oder mit der Station kollidieren.

□

Unter der Geometrie der Station verstehen wir die geometrische Beschreibung aller Hindernisse. Die ursprüngliche Formulierung ergibt sich als Spezialfall bei leerer Geometrie der Station. Wir verzichten auf dynamische Hindernisse.

1.2.3 Schweißzeiten

Anstatt gleiche Bearbeitungszeiten für Schweißpunkte gleichen Typs anzunehmen, verallgemeinern wir an dieser Stelle und geben jedem Schweißpunkt eine eigene Schweißzeit vor. Die ursprüngliche Situation ergibt sich als Spezialfall mit identischen Schweißzeiten.

1.2.4 Erweiterte Problembeschreibung

Zusammen mit den vorgestellten Erweiterungen ergibt sich die nachfolgende, erweiterte Problembeschreibung:

Definition 1.4 (Erweiterte Problembeschreibung) Gegeben eine Station, beschrieben durch ihre Geometrie und Anzahl, Positionen, Ausrichtungen, Geometrie und Kenndaten ihrer Roboter (baugleichen Typs), einen Sicherheitsabstand $s \in \mathbb{R}_{\geq 0}$, eine Menge von Schweißpunkten und deren Schweißzeiten sowie eine Menge von Nebenbedingungen. Finde eine Ansteuerung der Roboter, welche die folgenden Bedingungen erfüllt:

(1) *Optimalität:* Die Gesamtbearbeitungszeit, d. h. die Zeit, die vergeht, bis alle Roboter ihre Arbeit beendet haben, muss minimal sein.

(2) *Vollständigkeit:* Am Ende müssen alle Schweißpunkte bearbeitet worden sein.

(3) *Kollisionsfreiheit:* Kein Roboter darf mit sich selbst kollidieren. Zu keinem Zeitpunkt darf ein Roboter zu einem anderen Roboter, zu dem Bauteil oder zu der Station einen Abstand kleiner oder gleich s haben.

(4) *Nebenbedingungen:* Die folgenden Arten von Nebenbedingungen sind zulässig:
   (a) *Reihenfolge:* Ein Schweißpunkt muss vor einem anderen bearbeitet werden.
   (b) *Ausschluss:* Ein Roboter darf einen Schweißpunkt nicht bearbeiten.

□

1.3 Schwierigkeiten

Wir geben eine Auswahl schwieriger Aspekte des Problems.

1.3.1 Gegenseitige Beeinflussung der Roboter

Die möglichen Wege eines Roboters zwischen zwei Schweißpunkten können sich in Abhängigkeit von den Bewegungen der anderen Roboter ändern, z. B. kann sich ein anderer Roboter zwischen die Schweißpunkte bewegen und muss dann umfahren werden. Die (möglichen) Wege zwischen den Schweißpunkten können sich also in Abhängigkeit von Teilen der Lösung ändern. Insbesondere kann der Weg eines Roboters von den Wegen aller anderen Roboter beeinflusst werden.

Der Weg eines Roboters hängt auch von denjenigen Bewegungen der anderen Roboter ab, die erst zu einem späteren Zeitpunkt beginnen (z. B. kann sich ein Roboter zu einem abseits gelegenen Schweißpunkt bewegen; nachdem er zu schweißen begonnen hat, fährt ein anderer Roboter zu einer Gruppe dazwischen liegender Schweißpunkte, bearbeitet diese und blockiert während dieser Zeit den ersten Roboter).

Die gegenseitige Beeinflussung wirkt sich sowohl auf den Optimierungsaspekt als auch auf die Wegplanung aus. Ihr entstammt ein wesentlicher Teil der Komplexität des Problems. Man beachte, dass sich die Roboter nur dann direkt gegenseitig beeinflussen können, wenn sie einander nahe sind.

1.3.2 Unterschiedliche Wege und Wegzeiten

Die Wege — und dadurch auch die Wegzeiten — zwischen zwei Schweißpunkten können sich für verschiedene Roboter unterscheiden. Dies gilt sowohl bei gleichem Abstand der jeweiligen Schweißpunkte zueinander (durch unterschiedliche Positionierungen der Schweißpunkte bezüglich der Roboter können unterschiedliche Bewegungen notwendig werden) als auch bei identischer Positionierung der Schweißpunkte zu den Robotern (es kann Hindernisse geben, die nur von einem der Roboter umfahren werden müssen).

1.3.3 Wegplanung und Ansteuerung

Um die Schweißpunkte zu bearbeiten, müssen die Roboter diese so anfahren, dass der Schweißpunkt in geeigneter Weise zwischen den Enden der Schweißzange zu liegen kommt. Dafür gibt es jedoch viele Möglichkeiten: Für jeden Roboter und jeden Schweißpunkt existieren, eingeschränkt durch Kollisionen, mehrere Einstellungen der Gelenke, welche die Schweißzange geeignet auf dem Schweißpunkt positionieren.[3] Für die Wege zwischen den Schweißpunkten gibt es, ebenfalls bis auf Kollisionen, nahezu beliebig viele Möglichkeiten (begrenzt durch die Genauigkeit der Robotersteuerung).

Will man eine Folge von Punkten (etwa von einem Schweißpunkt zu einem anderen über Zwischenpunkte) entlang der sie verbindenden Geradenstücke abfahren, so muss der Roboter an jedem dieser Punkte zum Stillstand kommen (ansonsten würde der Roboter entweder einen Punkt nicht erreichen oder sich zeitweise nicht auf den Geradensegmenten befinden). Man verwendet daher „geschwungenere“ Bahnen (z. B. indem man die Geradenstücke auf geeignete Weise durch polynomielle Kurven verbindet); dadurch lässt sich die Wegzeit verringern, da kein Abbremsen mehr nötig ist. Der kürzeste Weg ist daher nicht immer der schnellste Weg. Es kommt erschwerend hinzu, dass die Steuereinheiten der Roboter nicht beliebig komplexe Bahnen ausführen können. [28]

1.3.4 Modellierung der Roboter

Für die Kollisionserkennung benötigt man ein dreidimensionales Modell des Werkstücks, der Umgebung und der Roboter. Letztere bestehen nicht nur aus starren Bauteilen: Die Kabel für die Stromversorgung, Ansteuerung etc. laufen, zu einem Kabelstrang zusammengefasst, am Roboterarm entlang. Dieser Kabelstrang ist beweglich und hängt durch („Affenschaukel“).

Bei neueren Robotermodellen wird der Kabelstrang weitgehend im Inneren des Roboterarms entlanggeführt und tritt nur an wenigen Stellen nach außen. Da der Kabelstrang durch seine Dicke nur begrenzt flexibel ist und es aus Sicherheitsgründen ohnehin nicht ratsam ist, andere Roboter in der Nähe des Kabelstrangs oder gar zwischen Roboterarm und Kabelstrang operieren zu lassen, kann man die austretenden Kabelstränge statisch modellieren, beispielsweise indem man für jedes austretende Kabelstück die konvexe Hülle über alle möglichen Stellungen des Roboterams bildet und diese als statisches Bauteil modelliert.

1.4 Formalisierung

Die Problemstellung enthält sowohl diskrete Elemente wie Schweißpunkte und deren Aufteilung auf die Roboter als auch kontinuierliche Elemente wie die Wege der Roboter im Raum. Wir beginnen mit der Vergabe einiger Bezeichner und geben dann zwei äquivalente Formalisierungen des Problems, wobei jeweils die kontinuierlichen bzw. die diskreten Aspekte im Vordergrund stehen. Die Wahl der Bezeichner ist zum Teil an die Terminologie des Sequencing & Scheduling [76] angelehnt. Tabelle 1 auf Seite xviii führt alle in diesem Kapitel vergebenen Bezeichner auf.

1.4.1 Konfigurationen der Roboter

Die Stellung eines Roboters im Raum wird (bei fester Position und Ausrichtung) durch die Winkelstellungen seiner Gelenke eindeutig bestimmt. Wir bezeichnen die Menge aller möglichen Winkelstellungen eines Roboters als den *Konfigurationsraum* $\mathcal{C}$ (engl. configuration space, C-space)

[3] Die Elektroden müssen senkrecht auf der Oberfläche des Werkstücks stehen; welche der beiden Elektroden sich auf welcher Seite befinden muss, steht fest. Die verschiedenen Positionierungsmöglichkeiten resultieren aus einer möglichen Drehung der Schweißzange um die Senkrechte durch den Schweißpunkt und den Freiheitsgraden des Schweißroboters (die gleiche Positionierung der Schweißzange kann durch verschiedene Stellungen des Roboters erreicht werden).

des Roboters. Die Teilmenge derjenigen Konfigurationen aus $\mathcal{C}$, die keine Kollision verursachen, bezeichnen wir als *freien* Konfigurationsraum.[4] Da die Roboter laut Annahme alle gleichen Typs sind, ist $\mathcal{C}$ für alle Roboter gleich.[5] Die Dimension von $\mathcal{C}$, auch Anzahl der Freiheitsgrade eines Roboters genannt, ist die Anzahl der zur Beschreibung der Stellung eines Roboters notwendigen Parameter, in unserem Fall die Anzahl der Gelenke, die wir mit $\kappa \in \mathbb{N}$ bezeichnen.

Beispiel 1.1 (Konfigurationsraum KR 150 K) Die nebenstehende Tabelle zeigt die zulässigen Bereiche für die Winkeleinstellungen der sechs Gelenke eines KR 150 K Roboters der *KUKA Roboter GmbH* [89]. Die Daten sind der Spezifikation [31] des Roboters entnommen. Der entsprechende Konfigurationsraum ist

$$\mathcal{C} = [-185,185] \times [-120,70] \times [-209,65] \times [-350,350] \times [-125,125] \times [-350,350] \subset \mathbb{R}^6.$$

Gelenk	Maximum	Minimum
1	−185	+185
2	−120	+70
3	−209	+65
4	−350	+350
5	−125	+125
6	−350	+350

Tabelle 1.1: Zulässige Winkelstellungen des KR 150 K

□

1.4.2 Roboter und Schweißpunkte

Wir bezeichnen die Anzahl der Roboter mit $m \in \mathbb{N}$ und den i-ten Roboter der Station mit M_i. Ein Roboter ist für uns ein 4-Tupel $M_i \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathcal{C} \times \mathcal{G}$, $1 \leq i \leq m$. Dabei steht die erste Komponente für die Position des Fußpunkts des Roboters (bezogen auf das Koordinatensystem der Station, siehe den nachfolgenden Abschnitt), die zweite Komponente für die Ausrichtung (Drehwinkel um jede Achse, siehe Fußnote auf Seite 2), die dritte Komponente für die *Startstellung* S_i und die vierte Komponente für die Geometriedaten des Roboters. Die Menge $\mathcal{G}$ bezeichnet die Menge aller möglichen Geometriedaten.

Da der Schwerpunkt dieser Arbeit auf dem Optimierungsaspekt liegt, gehen wir auf die formale Definition der Geometriedaten sowie die Programmierung der Roboter nicht näher ein. Man denke bei den Geometriedaten beispielsweise an eine Folge einfacher dreidimensionaler geometrischer Objekte (Kugeln, Kapseln, Lozenges, Zylinder, Ellipsoide, achsenausgerichtete Quader, orientierte Quader) [3] mit Positionsangaben im Roboterkoordinatensystem. Eine andere Möglichkeit zur Spezifikation der Robotergeometrie ist die Beschreibung der Oberfläche eines Roboters durch eine Folge von konvexen Polygonen wie z. B. Dreiecken. Abschnitt 2.1 geht im Rahmen des Literaturüberblicks näher auf die Repräsentation von Objektgeometrien ein.

Die Anzahl der Schweißpunkte bezeichnen wir mit $n \in \mathbb{N}$, den i-ten Schweißpunkt mit $J_i \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}_{>0}$ und die Menge aller Schweißpunkte mit $J := \{J_i \mid 1 \leq i \leq n\}$. Die erste Komponente eines Schweißpunkts bezeichnet seine Position im Fahrzeugkoordinatensystem, die dritte Komponente seine Schweißzeit $t_i \in \mathbb{R}_{>0}$. Bei der zweiten Komponente handelt es sich um einen Vektor, der, ausgehend von der Position des Schweißpunkts, senkrecht auf der Oberfläche des Werkstücks

[4] Zur Berücksichtigung von Kollisionen des Roboters mit sich selbst und mit statischen Hindernissen (der Geometrie der Station) genügt es, $\mathcal{C}$ entsprechend einzuschränken. Dabei ist zu beachten, dass man bei Tests auf Kollisionsfreiheit eines Roboters mit sich selbst aneinander angrenzende, d. h. durch ein Gelenk verbundene Achsen im Normalfall nicht auf Kollision prüft.

Kollisionen mit anderen, sich bewegenden Robotern lassen sich auf diese Weise nicht berücksichtigen, da sich die nicht kollisionsfreien Stellungen eines Roboters mit den Bewegungen der anderen Roboter ändern. Um Kollisionen mit anderen Robotern über den freien Konfigurationsraum zu berücksichtigen, kann man alle Roboter zusammen als einen — räumlich nicht zusammenhängenden — Roboter betrachten. Die Dimension des entsprechenden Konfigurationsraums ist die Summe der Dimensionen der einzelnen Konfigurationsräume, $m \cdot \kappa$. Kollisionen zwischen einzelnen Robotern werden dadurch zu Kollisionen eines Roboters mit sich selbst.

[5] Während die Roboter alle gleich sind, gilt dies nicht für die Schweißzangen. Da die Schweißzangen jedoch über keine Freiheitsgrade verfügen, bleibt der Konfigurationsraum $\mathcal{C}$ davon unberührt.

steht. Er dient zur korrekten Positionierung der Schweißzange (die Elektroden müssen entlang dieses Vektors ausgerichtet sein; siehe Fußnote auf Seite 5); die Richtung des Vektors bestimmt, welche Elektrode sich auf welcher Seite des Werkstücks befinden muss. Die Menge aller Stellungen eines Roboters, die zu einer korrekten Positionierung seiner Schweißzange auf einem bestimmten Schweißpunkt führen, bezeichnen wir mit

$$\mathcal{V}_{i,j} := \{ s \in \mathcal{C} \mid \text{Die Schweißzange von } M_i \text{ ist auf } J_j \text{ positioniert} \}.$$

1.4.3 Koordinatensysteme und Abstände

Positionsangaben können sich auf unterschiedliche Koordinatensysteme beziehen. Das Weltkoordinatensystem ist üblicherweise das Koordinatensystem der Werkshalle; für unsere Zwecke genügt jedoch das Koordinatensystem der Station. Das Werkstück hat ebenfalls ein eigenes Koordinatensystem, das sogenannte Fahrzeugkoordinatensystem, ebenso jeder Roboter.

Den Abstand zwischen zwei Robotern M_i und M_j definieren wir als die kleinste Entfernung zwischen einem Bauteil von M_i und einem Bauteil von M_j.[6] Er lässt sich unter Zuhilfenahme von Position, Ausrichtung, Stellung und Geometrie der beiden Roboter berechnen. Wir bezeichnen ihn mit $d_{i,j} : \mathcal{C}^2 \to \mathbb{R}_{\geq 0}$ (Position, Ausrichtung und Geometrie sind fest; bei nicht-leerem Schnitt der beiden Roboter ist der Abstand 0). Den Abstand von M_i zum Werkstück und zur Station[7] bezeichnen wir analog mit $d_i : \mathcal{C} \to \mathbb{R}_{\geq 0}$.

1.4.4 Nebenbedingungen

Wir führen zwei Arten von Nebenbedingungen ein, eine in Bezug auf die Reihenfolge der Bearbeitung von Schweißpunkten und eine bezüglich des Ausschlusses von Robotern von der Bearbeitung mancher Schweißpunkte. Die Nebenbedingungen bezüglich der Reihenfolge der Bearbeitung resultieren aus externen produktionstechnischen Anforderungen. Wir bezeichnen die Menge dieser Nebenbedingungen mit

$$C_S \subseteq \left\{ (i,j) \in \mathbb{N}^2 \;\middle|\; 1 \leq i,j \leq n \right\},$$

wobei $(i,j) \in C_S$ angibt, dass Schweißpunkt J_i vor Schweißpunkt J_j bearbeitet werden muss, d. h. der Zeitpunkt, zu dem die Bearbeitung von J_i beendet wird, muss vor dem Zeitpunkt liegen, zu dem die Bearbeitung von J_j beginnt. Die Relation C_S definiert eine irreflexive partielle Ordnung[8] auf den Schweißpunkten.

Die Nebenbedingungen bezüglich des Ausschlusses von Robotern von der Bearbeitung mancher Schweißpunkte resultieren aus stationsinternen Anforderungen. Die hauptsächlichen Faktoren sind dabei Erreichbarkeit (ein Roboter kann einen Schweißpunkt aufgrund mangelnder Reichweite oder wegen Hindernissen nicht erreichen) und unterschiedliche Arten von Schweißzangen (nicht jede Schweißzange kann jeden Punkt schweißen). Wir bezeichnen die Menge dieser Nebenbedingungen mit

$$C_E \subseteq \left\{ (i,j) \in \mathbb{N}^2 \;\middle|\; 1 \leq i \leq m, 1 \leq j \leq n \right\},$$

wobei $(i,j) \in C_E$ bedeutet, dass der Roboter M_i den Schweißpunkt J_j nicht bearbeiten darf. Das auf den Schweißpunkt J_j beschränkte Komplement von C_E, d. h. die Menge der Roboter, die den Schweißpunkt J_j bearbeiten dürfen, nennen wir $a_j := \{1, \ldots, m\} \setminus \{ k \mid (k,j) \in C_E \}$.

[6] Ist die Geometrie der Roboter als eine Menge von Polygonen gegeben (siehe Abschnitt 2.1) und bezeichnen A und B die Mengen der gemäß Position, Ausrichtung und Stellung der jeweiligen Roboter transformierten Eckpunkte der Polygone, dann gilt $d_{i,j} = \min \{ d(x,y) \mid x \in A, y \in B \}$, wobei $d : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}_{\geq 0}$ den üblichen euklidischen Abstand im dreidimensionalen Raum bezeichnet.

[7] Da das Werkstück während der Bearbeitung nicht bewegt wird, kann es bezüglich der Kollisionserkennung als statisches Hindernis betrachtet werden. Somit ist für Kollisionserkennung und Abstandsmessung eine Unterscheidung zwischen Station und Werkstück nicht nötig.

[8] Die Relation C_S ist irreflexiv $(i,i) \notin C_S$, asymmetrisch $(i,j) \in C_S \Rightarrow (j,i) \notin C_S$ und transitiv $(i,j) \in C_S \wedge (j,k) \in C_S \Rightarrow (i,k) \in C_S$.

1.4.5 Kontinuierliche Formulierung

Wir geben eine formale Definition des Schweißzellenproblems (WCP, engl. Welding Cell Problem) auf Basis der von den Robotern im Raum ausgeführten Bewegungen, ausgehend von der erweiterten Problembeschreibung in Definition 1.4. Dabei konzentrieren wir uns wie bisher auf den Optimierungsaspekt; bezüglich der Kollisionserkennung beschränken wir uns auf die Verwendung der Metriken $d_{i,j}$ und d_i.

Gemäß Definition 1.4 ist eine Ansteuerung der Roboter gesucht, die gewissen Bedingungen genügt. Eine solche Ansteuerung hat eindeutige Bewegungen und damit *Trajektorien*[9] der Roboter zur Folge. Um von den konkreten Schnittstellen der einzelnen Robotersteuerungen zu abstrahieren, wählen wir als Lösungsraum nicht die Ansteuerungen der Roboter. Stattdessen nehmen wir an, dass wir für eine gegebene Trajektorie eine passende Ansteuerung bestimmen können, sofern diese existiert, d. h. falls die Trajektorie durch den Roboter realisiert werden kann.[10] Weiterhin nehmen wir an, dass wir zwischen realisierbaren und nicht realisierbaren Trajektorien unterscheiden können.

Die Position eines Roboters im Raum wird durch die Winkelstellungen seiner Gelenke eindeutig bestimmt. Zur Beschreibung einer Trajektorie genügt daher die Angabe der Winkelstellungen des Roboters zu jedem Zeitpunkt. Eine solche Funktion, die jedem Zeitpunkt ein Tupel von Winkelstellungen zuordnet, nennen wir *Bewegung*. Wir wählen als Lösungsraum die Menge aller Bewegungen, für die eine Ansteuerung existiert und bei denen der Roboter nicht mit sich selbst kollidiert:

$$\mathcal{M} := \left\{ f : \mathbb{R}_{\geq 0} \to \mathcal{C} \;\middle|\; \begin{array}{l} \text{es gibt eine Ansteuerung für } f \;\wedge\; \forall t \in \mathbb{R}_{\geq 0} : \text{der Ro-} \\ \text{boter kollidiert in Stellung } f(t) \text{ nicht mit sich selbst} \end{array} \right\}.$$

Die Funktionen in $\mathcal{M}$ sind in jeder Komponente gleichmäßig stetig.[11] Die erste Ableitung liefert die Geschwindigkeit der Winkeländerung, die zweite Ableitung die Beschleunigung.

Definition 1.5 (WCP, Kontinuierliche Formulierung) Gegeben Station, Schweißpunkte und Nebenbedingungen wie sie in den vorherigen Abschnitten definiert wurden.

Finde m Bewegungen $f_1, \ldots, f_m \in \mathcal{M}$ und $t_{\max} \in \mathbb{R}_{>0}$ sowie n Schweißzeitpunkte

$$s_j \in \,]0, t_{\max} - t_j[\,\subset \mathbb{R}_{>0}, \quad 1 \leq j \leq n,$$

so dass die folgenden Bedingungen erfüllt sind:

(1) *Optimalität:* Die Gesamtbearbeitungszeit muss minimal sein:

$$t_{\max} = \min \left\{ t \in \mathbb{R}_+ \;\middle|\; \begin{array}{l} \exists g_1, \ldots, g_m \in \mathcal{M}\, \exists r_1 \in \,]0, t - t_1[, \ldots, r_n \in \,]0, t - t_n[: \\ g_1, \ldots, g_m, r_1, \ldots, r_n, t \text{ erfüllen die Bedingungen 2,3,4,5} \end{array} \right\}.$$

(2) *Vollständigkeit:* Am Ende müssen alle Schweißpunkte bearbeitet worden sein:

$$\forall j \in \{1, \ldots, n\}\, \exists i \in a_j : f_i([s_j, s_j + t_j]) = v \in \mathcal{V}_{i,j}.$$

Die Bedingung stellt sicher, dass es für jeden Schweißpunkt einen Roboter gibt, der ihn zum vorgesehenen Zeitpunkt bearbeiten kann.

[9] Von *lat.* traicere, hinüberwerfen. In der Physik versteht man unter einer Trajektorie den Weg eines sich bewegenden Objekts im Raum.

[10] Ohne Berücksichtigung von Kollisionen, sondern ausschließlich auf die Ansteuerbarkeit bezogen. Die Einschränkungen ergeben sich durch die Schnittstelle der Robotersteuerung und physikalische Begrenzungen wie mechanische Belastbarkeit der Bauteile, auftretende Beschleunigungskräfte, Spezifikationen der Motoren usw.

[11] Durch die endliche Leistungsfähigkeit von Motoren lässt sich für jedes $\epsilon > 0$ ein $\delta > 0$ finden, so dass die Winkelstellung des Gelenks sich in Zeit δ um nicht mehr als ϵ ändern kann.

(3) *Kollisionsfreiheit:* Kein Unterschreiten des Sicherheitsabstandes zwischen Robotern, Bauteil und Station:

$$\forall t \in [0, t_{\max}]\ \forall i, j \in \{1, \dots, m\}, i \neq j : \Big(d_{i,j}\big(f_i(t), f_j(t)\big) > s\Big) \wedge \Big(d_i\big(f_i(t)\big) > s\Big).$$

(4) *Nebenbedingungen:*

(a) *Reihenfolge:* Manche Schweißpunkte müssen vor anderen bearbeitet werden:

$$\forall (i, j) \in C_S : s_i + t_i < s_j.$$

(b) *Ausschluss:* Manche Roboter dürfen manche Schweißpunkte nicht bearbeiten.
Die Sicherstellung der Ausschluss-Nebenbedingungen erfolgt implizit durch die Verwendung der a_j in Punkt 2.

(5) *Sonstiges:* Jeder Roboter beginnt und beendet seine Bewegung in seiner Ausgangsstellung:

$$\forall i \in \{1, \dots, m\} : f_i(0) = f_i(t_{\max}) = S_i \ \wedge \ f_i'(0) = f_i'(t_{\max}) = \vec{0}.$$

□

Gibt es eine Ansteuerung, welche die Bedingungen 2 bis 5 erfüllt, dann existiert das Minimum in Bedingung 1 aufgrund der Einschränkung auf ansteuerbare Funktionen. In Bedingung 2 wird sichergestellt, dass sich die Schweißzange während des Schweißens nicht bewegt. Die Berücksichtigung von Kollisionen des Roboters mit sich selbst erfolgt durch die Verwendung von $\mathcal{M}$.

Während die Wahl von $\mathcal{G}$ für Definition 1.5 transparent ist (da nur $d_{i,j}$ und d_i verwendet werden), ist die Wahl von $\mathcal{M}$ für die Arbeit mit Definition 1.5 wesentlich. Beispiele für die Wahl von $\mathcal{M}$ werden wir in Abschnitt 3.3 kennen lernen.

1.4.6 Diskrete Formulierung

In Definition 1.5 stehen die als kontinuierliche Funktionen modellierten Bewegungen der Roboter im Vordergrund. Geht man stattdessen von der Aufteilung der Schweißpunkte auf die Roboter und der Festlegung von Bearbeitungsreihenfolgen aus, erhält man eine äquivalente Formulierung, bei der die diskreten Aspekte mehr im Vordergrund stehen.

Wir beobachten zuerst, dass die Roboter an den Schweißzeitpunkten s_j still stehen (Bedingung 2 in Definition 1.5). Da die Lösungen kollisionsfrei sind, ist durch die Schweißzeitpunkte eindeutig festgelegt, welcher Roboter welche Schweißpunkte bearbeitet (zu einem Schweißzeitpunkt kann immer nur ein Roboter auf einem Schweißpunkt positioniert sein) und in welcher Reihenfolge dies geschieht (Reihenfolge der Schweißzeitpunkte eines Roboters). Für einen Roboter ist damit festgelegt, in welcher Reihenfolge er sich von Schweißpunkt zu Schweißpunkt bewegt. Wir teilen die Bewegungen der Roboter in kleinere Teilbewegungen von Schweißpunkt zu Schweißpunkt auf und erhalten so die gewünschte Formulierung.

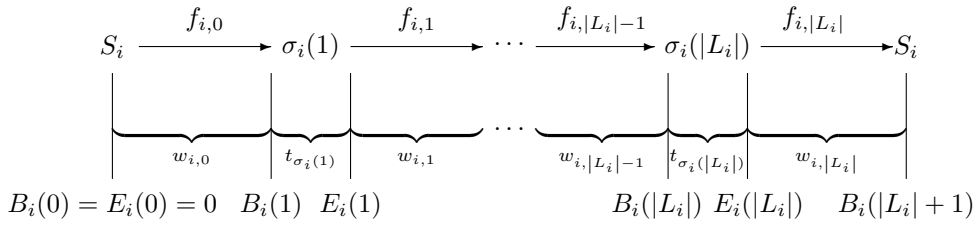


Abbildung 1.1: Zusammensetzung des Weges von M_i in Definition 1.6

Definition 1.6 (WCP, Diskrete Formulierung) Gegeben Station, Schweißpunkte und Nebenbedingungen, wie sie in den vorherigen Abschnitten definiert wurden.

Finde eine Partitionierung der Schweißpunkte in m Mengen

$$L_1, \ldots, L_m \subseteq \{1, \ldots, n\}$$

und eine Anordnung jeder dieser Mengen

$$\sigma_i : \{1, \ldots, |L_i|\} \to L_i, \quad \sigma_i \text{ bijektiv}$$

sowie m Folgen von Bewegungen mit dazugehörigen Wegzeiten

$$f_{i,0}, \ldots, f_{i,|L_i|} \in \mathcal{M}, \quad w_{i,0}, \ldots, w_{i,|L_i|} \in \mathbb{R}_{>0}, \quad 1 \leq i \leq m,$$

und für jeden Schweißpunkt eine geeignete Schweißstellung

$$v_{i,1} \in \mathcal{V}_{i,\sigma_i(1)}, \ldots, v_{i,|L_i|} \in \mathcal{V}_{i,\sigma_i(|L_i|)}, \quad 1 \leq i \leq m,$$

so dass die nachfolgenden Bedingungen erfüllt sind. Dabei bezeichne

$$B_i : \{0, \ldots, |L_i| + 1\} \to \mathbb{R}_{>0}, \quad B_i(j) := \sum_{k=0}^{j-1} w_{i,k} + \sum_{k=1}^{j-1} t_{\sigma_i(k)}$$

die Anfangszeit der Bearbeitung des j-ten Schweißpunktes in L_i und

$$E_i : \{0, \ldots, |L_i|\} \to \mathbb{R}_{>0}, \quad E_i(j) := \begin{cases} B_i(j) + t_{\sigma_i(j)} & \text{falls } 1 \leq j \leq |L_i|\,, \\ 0 & \text{falls } j = 0 \end{cases}$$

deren Endzeit. Weiterhin sei

$$t_{\max} := \max\Big\{ B_i\big(|L_i| + 1\big) \Big| 1 \leq i \leq m \Big\} \tag{1.1}$$

die Gesamtbearbeitungszeit und

$$f_i(t) := \begin{cases} f_{i,k}(t - E_i(k)) & \text{falls } M_i \text{ sich von } J_{\sigma_i(k)} \text{ nach } J_{\sigma_i(k+1)} \text{ bewegt, d. h.} \\ & \exists k \in \{0, \ldots, |L_i|\} : E_i(k) \leq t \leq B_i(k+1) \\ v_{i,k} & \text{falls } M_i \text{ den Schweißpunkt } J_{\sigma_i(k)} \text{ bearbeitet, d. h.} \\ & \exists k \in \{1, \ldots, |L_i|\} : B_i(k) < t \leq E_i(k) \\ 0 & \text{falls } t > t_{\max} \end{cases}$$

die durch die einzelnen Teilbewegungen induzierte Gesamtbewegung von M_i.

(1) *Optimalität:* Die Gesamtbearbeitungszeit muss minimal sein:

$$t_{\max} = \min \left\{ t \in \mathbb{R}_{>0} \;\middle|\; \begin{array}{l} \exists f'_{1,0}, \ldots, f'_{m,|L_m|} \in \mathcal{M}\; \exists w'_{1,0}, \ldots, w'_{m,|L_m|} \in \mathbb{R}_{>0} : \\ \text{die Bedingungen 2,3,4,5 sind erfüllt,} \\ \text{wobei } t \text{ gemäß Gleichung 1.1 berechnet wird.} \end{array} \right\}.$$

(2) *Vollständigkeit:* Am Ende müssen alle Schweißpunkte bearbeitet worden sein.

Die Bearbeitung aller Schweißpunkte ist implizit durch die Partitionierung sichergestellt.

(3) *Kollisionsfreiheit:* Kein Unterschreiten des Sicherheitsabstandes zwischen Robotern, Bauteil und Station:

$$\forall t \in [0, t_{\max}]\ \forall i, j \in \{1, \ldots, m\}, i \neq j : \Big(d_{i,j}\big(f_i(t), f_j(t)\big) > s\Big) \wedge \Big(d_i\big(f_i(t)\big) > s\Big).$$

(4) *Nebenbedingungen:* Es bezeichne $L : \{1, \ldots, n\} \to \{1, \ldots, m\}$ mit $L(j) = i$ falls $j \in L_i$ die Funktion, die einem Schweißpunkt seine Partition zuordnet und σ^{-1} die Umkehrfunktion zu σ.

(a) *Reihenfolge:* Manche Schweißpunkte müssen vor anderen bearbeitet werden:

$$\forall (i, j) \in C_S : E_{L(i)}(\sigma^{-1}_{L(i)}(i)) < B_{L(j)}(\sigma^{-1}_{L(j)}(j)).$$

(b) *Ausschluss:* Manche Roboter dürfen manche Schweißpunkte nicht bearbeiten:

$$\forall (i, j) \in C_E : j \notin L_i.$$

(5) *Konsistenz:* Anfang und Ende der Teilbewegungen sind miteinander, den Schweißstellungen und den Startstellungen kompatibel:

$$\begin{aligned} &\forall i \in \{1, \ldots, m\} : \forall j \in \{1, \ldots, |L_i|\} : \\ &\qquad f_{i,j}(0) = f_{i,j-1}(w_{i,j-1}) = v_{i,j} \quad \wedge \quad f'_{i,j}(0) = f'_{i,j-1}(w_{i,j-1}) = 0, \\ &\forall i \in \{1, \ldots, m\} : f_{i,0}(0) = f_{i,|L_i|}(w_{i,|L_i|}) = S_i \quad \wedge \quad f'_{i,0}(0) = f'_{i,|L_i|}(w_{i,|L_i|}) = 0. \end{aligned}$$

□

In der diskreten Formulierung wird der Weg eines Roboters durch die bearbeiteten Schweißpunkte und die dazwischen liegenden Wegstücke bestimmt. Die Zusammensetzung der Bewegung von Roboter M_i und die einzelnen Bearbeitungszeitpunkte werden in Abbildung 1.1 noch einmal grafisch dargestellt. Man beachte, dass M_i im Fall $L_i = \emptyset$ keine Bewegung ausführt.

1.4.7 Vergleich und Anmerkungen

Die beiden Formulierungen des WCP sind äquivalent in dem Sinne, dass jede Lösung der kontinuierlichen Formulierung einer Lösung der diskreten Formulierung entspricht und umgekehrt.[12] Sie unterscheiden sich im Wesentlichen in der Darstellung der Lösung und stellen so jeweils andere Aspekte des Problems in den Vordergrund. Obwohl die diskrete Formulierung auf den ersten Blick komplizierter wirkt als die kontinuierliche Formulierung, ist sie für gewisse Ansteuerungsmodelle natürlicher (siehe Abschnitte 3.3.1 und 3.3.2). Bei beiden Formulierungen wurde bewusst von der konkreten Ansteuerung der Roboter abstrahiert, um so weit wie möglich unabhängig von den Besonderheiten der einzelnen Robotersteuerungen zu bleiben.

[12] Zwei Lösungen entsprechen einander, wenn sie den gleichen Wert für $t_{\max}$ verwenden und die Bewegungen $f_1, \ldots, f_m$ der Roboter im Intervall $[0, t_{\max}]$ identisch sind. Tatsächlich gibt es zu jeder kontinuierlichen Lösung genau eine ihr entsprechende diskrete Lösung, zu jeder diskreten Lösung aber existieren unendlich viele ihr entsprechende kontinuierliche Lösungen. Dies liegt daran, dass in der diskreten Formulierung $f_i(t) = 0$ für $t > t_{\max}$ gesetzt wird, während in der kontinuierlichen Formulierung keine diesbezüglichen Forderungen an die f_i gestellt werden. Bei Bedarf ließe sich durch die zusätzliche Bedingung $\forall i \in \{1, \ldots, m\} \forall t > t_{\max} : f_i(t) = 0$ unter Punkt 5 in Definition 1.5 erreichen, dass jeder diskreten Lösung genau eine kontinuierliche Lösung entspricht, wodurch eine 1–1-Korrespondenz der Lösungen von kontinuierlicher und diskreter Formulierung gegeben wäre.

1.5 Weiterführendes

Die in diesem Kapitel erfolgte Begriffsbildung stellt zusammen mit den Definitionen 1.5 und 1.6 die Grundlage der weiteren Kapitel dar. In der Praxis existieren zahlreiche weitere Varianten des Schweißzellenproblems sowie verwandte Problemstellungen, von denen wir einige vorstellen.

Mehrere Robotertypen Die Roboter einer Station dürfen unterschiedlichen Typs sein. Dadurch benötigt jeder Roboter seinen eigenen Konfigurationsraum und damit seine eigene Menge zulässiger Bewegungen; die Auswirkungen betreffen hauptsächlich die Modellierung der Roboter und die Kollisionserkennung. In der Praxis wird meist das gleiche Robotermodell mit unterschiedlichen Schweißzangen eingesetzt (was bei unserer Modellierung aufgrund der unterschiedlichen Geometrien der Schweißzangen unterschiedliche Robotertypen erfordert). Häufig kommen auch geringfügig modifizierte Versionen eines Robotermodells zum Einsatz, z. B. mit Armverlängerungen oder einer zusätzlichen Linearachse.

Werkzeugwechsel Es gibt unterschiedliche Schweißzangen für verschiedene Arten von Schweißpunkten. Bei der Fertigung von Kleinserien ist nicht immer für jeden Typ von Schweißzange ein eigener Roboter vorhanden, sondern der Roboter kann die Schweißzange wechseln. Dazu fährt er in die Ausgangsstellung zurück, wechselt die Schweißzange (dies kann manuell oder automatisch geschehen) und fährt dann mit der Arbeit fort.

Bei feststehender Reihenfolge der Zangenwechsel können wir dies in unserem Modell wie folgt berücksichtigen: Für jeden Schweißzangentyp erstellen wir einen neuen Schweißpunkt, wobei Position und Ausrichtung des Schweißpunktes so zu wählen sind, dass der Roboter[13] beim Anfahren des Punktes in Startstellung steht. Die Schweißzeit setzen wir auf die für den Zangenwechsel benötigte Zeit.[14] Durch entsprechende Reihenfolge-Nebenbedingungen lässt sich erreichen, dass alle einem Zangentyp zugeordneten Schweißpunkte vor dem neuen, den Zangenwechsel repräsentierenden Schweißpunkt bearbeitet werden müssen.

Steht die Reihenfolge der Zangenwechsel nicht fest, können mehrere Optimierungsläufe für unterschiedliche Reihenfolgen der Zangenwechsel gefahren werden. Bei k für einen Roboter vorgesehenen Zangentypen gibt es $k!$ mögliche Reihenfolgen für die Zangenwechsel, was bei sehr wenigen Zangenwechseln ($k \leq 4$) noch machbar ist.

Fliegender Wechsel Die Roboter werden nach erfolgter Bearbeitung des Werkstücks nicht wieder in die Ausgangsstellung zurückgefahren; dadurch gehen die Bearbeitungszyklen nahtlos ineinander über, wobei Anfangs- und Endstellung identisch sein müssen, um die räumliche Kontinuität zu wahren. Dieses Vorgehen entspricht einer Aufnahme der Ausgangsstellung in den Lösungsraum. An- und Abtransport des Werkstücks sollten in die Problemstellung miteinbezogen werden. Durch die eingesparten Bewegungen kann sich die Bearbeitungszeit verkürzen.

Feste Schweissvorrichtung Gelegentlich ist die Schweißvorrichtung fest installiert, so dass der Roboter das Werkstück erst aufnehmen, dann für jeden Schweißpunkt passend an der Schweißvorrichtung positionieren und es anschließend wieder ablegen muss. Aufgabenstellungen dieser Art erfordern meist auch die Berücksichtigung von An- und Abtransport des Werkstücks.

Hat man sich für eine Modifikation der Problemdefinition entschieden, um die Beweglichkeit des Werkstücks zu berücksichtigen, lässt sich die korrekte Positionierung eines Roboters für Aufnahme, Positionierung und Ablage des Werkstücks durch geeignete Wahlen der $\mathcal{V}_{i,j}$ erreichen. Sollen Roboter das Werkstück einander übergeben, kann dies durch einen neuen Typ von Nebenbedingungen der Form „Schweißpunkt J_i und Schweißpunkt J_j müssen gleichzeitig bearbeitet werden“ erreicht werden.

[13] Wir gehen davon aus, dass es für jeden Schweißzangentyp nur einen Roboter gibt, der damit ausgerüstet ist.

[14] Bei manuellen Zangenwechseln ist hier ein experimentell zu bestimmender Wert für die benötigte Zeit einzusetzen. Wird dieser beim Wechsel der Zange überschritten, muss die Station evtl. angehalten werden (Kollisionsgefahr).

Laserschweissen Statt mit elektrischem Strom wird die Schweißstelle mit einem Laser erhitzt. Man unterscheidet Wärmeleitungsschweißen (Erhitzung der Materialoberfläche durch Absorption des Laserstrahls; die Einschweißtiefe beträgt typischerweise weniger als 1 mm) und Tiefschweißen (Erzeugung einer Dampfkapillare durch lokale Verdampfung des Materials mittels des Lasers; bei Stahl sind Einschweißtiefen bis zu 25 mm möglich). Das Licht wird dabei vom eigentlichen Laser zum Effektor des Roboters geleitet, z. B. mittels Lichtleitfasern bei Nd:YAG-Lasern[15] oder flexibler Spiegelführung bei CO_2-Lasern. Beim Laserschweißen wird mit Schweißnähten an Stelle von Schweißpunkten gearbeitet.

Ein Vorteil des Laserschweißens ist die bessere Erreichbarkeit der Schweißstelle, da im Gegensatz zum elektrischen Widerstandsschweißen nur eine einseitige Zugänglichkeit benötigt wird. Ein Nachteil sind die höheren Kosten; oft lohnt sich das Laserschweißen erst ab einer Auslastung des Lasers von mindestens 90%. Beim Laserschweißen müssen sich daher meist mehrere Roboter einen Laser teilen, wodurch immer nur einer dieser Roboter gleichzeitig schweißen kann.

Der Übergang zum Laserschweißen wirkt sich auf die Modellierung der Schweißstellen aus, da diese nunmehr eine räumliche Ausdehnung besitzen. Die Verwendung eines Lasers durch mehrere Roboter ist in der Formulierung des Optimierungsproblems zu berücksichtigen, z. B. durch Indikatorvariablen für jeden Laser, eine gegenseitige Ausschlussmatrix für die Roboter oder Nebenbedingungen der Form „Roboter M_i und Roboter M_j dürfen nicht gleichzeitig schweißen“.

Andere Arbeitsaufgaben Die Betrachtungen in diesem und den folgenden Kapiteln lassen sich auf andere, ähnlich gelagerte Aufgabenstellungen übertragen. Ein Beispiel ist die automatisierte Verschraubung von Werkstücken. Allgemein scheinen Aufgabenstellungen geeignet, bei denen mehrere mit einem Effektor versehene Knickarmroboter eine in der Reihenfolge ihrer Ausführung nicht vollständig determinierte Menge von Aufgaben an einem Werkstück durchzuführen haben.

Weitere Optimierungskriterien Wir interessieren uns für zeitoptimale, kollisionsfreie Wege. Es gibt noch andere mögliche Optimierungskriterien; so führen etwa abrupte Beschleunigungen und ruckartige Bewegungen (wie sie z. B. auftreten, wenn man eine Folge von Punkten mit maximaler Beschleunigung entlang der sie verbindenden Geraden abfährt) zu Erschütterungen und über die so entstehenden Resonanzfrequenzen zu erhöhtem Verschleiß der Roboter. [27]

Ein weiteres mögliches Kriterium ist die Sicherheit gegenüber Verzögerungen bei der Ausführung. Dabei nehmen wir an, dass es zu zufälligen zeitlichen Verzögerungen bei der Ausführung von Bewegungen und Schweißvorgängen durch die Roboter kommen kann. Diesen kann man z. B. durch die Maximierung des zeitlichen Abstands, mit dem die Roboter bei ordnungsgemäßem Bahnverlauf räumlich gemeinsam genutzte Gebiete durchqueren, begegnen. Eine weitere Möglichkeit ist die Minimierung der Kollisionswahrscheinlichkeit auf Basis eines stochastischen Modells der Verzögerungen.

Die Berücksichtigung mehrerer Optimierungskriterien führt zur Optimierung mit Mehrfachkriterien (siehe Seite 23).

Verschiedene Lösungen Anstatt nach einer einzigen, optimalen Lösung zu suchen, kann man versuchen, mehrere gute Lösungen zu bestimmen, die sich in ihrer Struktur möglichst stark unterscheiden. Diese können dann als Vorlage für einen menschlichen Planer dienen. Die Forderung nach einer möglichst unterschiedlichen Struktur der Lösungen dient dazu, die Vorlage vieler fast identischer Lösungen zu vermeiden. Dieser Ansatz erfordert ein Maß für die Ähnlichkeit von Lösungen.

Stationsplanung Ist die Station noch in der Planung oder baulich veränderbar, so kann man Anzahl, Position und Ausrichtung der Roboter sowie Auswahl, Anzahl und Zuweisung der Schweißzangentypen in die Optimierung einbeziehen. Als Optimierungskriterium kommt neben der Bearbeitungszeit z. B. die Anzahl der Roboter in Betracht. Hierbei sind die verschiedenen zu fertigenden

[15] Die Abkürzung steht für Neodym dotiertes Yttrium-Aluminium-Granat, das verwendete Lasermedium.

Werkstücke zu berücksichtigen und Reserven einzuplanen, da für gewöhnlich Anzahl und Position der Schweißpunkte über die Lebenszeit der Station hinweg immer wieder (geringfügigen) Änderungen unterworfen sind. Diese Aufgabenstellung würde das Schweißzellenproblem als Teilproblem enthalten.

Kapitel 2

Klassifikation und Ansätze

Wir ordnen die einzelnen Aspekte des Schweißzellenproblems in die bestehende Forschungslandschaft ein, d. h. wir identifizieren relevante Forschungsgebiete, stellen einen Bezug zu bekannten Problemen her und geben einen Überblick über bestehende Verfahren. Wir behalten dazu die Aufteilung in Kollisionserkennung, Wegplanung, Ansteuerung und Optimierung bei; der Optimierungsaspekt steht dabei weiterhin im Vordergrund.

2.1 Kollisionserkennung

Der Aspekt der Kollisionserkennung innerhalb des Schweißzellenproblems ist ein Spezialfall des allgemeinen Problems der Kollisionserkennung. Dieses gehört zum Gebiet der grafischen Datenverarbeitung, wo es vor allem in (interaktiven) dreidimensionalen Anwendungen wie Computerspielen, CAD/CAM (engl. Computer Aided Design / Computer Aided Manufacturing) und virtueller Realität auftritt. Weitere Anwendungen finden sich u. a. in der Robotik und der automatisierten Fertigung. Es wurde von verschiedenen Perspektiven wie rechenbetonter Geometrie (engl. computational geometry), geometrischer Modellierung und kollisionsfreier Pfadplanung heraus ausführlich studiert.

2.1.1 Gegenstand des Gebiets

Beim allgemeinen Problem der Kollisionserkennung sind geometrische Modelle sich bewegender Objekte gegeben. Man interessiert sich dafür, zu welchen Zeitpunkten der Schnitt zwischen diesen Objekten nicht leer ist, oft auch dafür, welche Teile der Objekte betroffen sind oder wie weit die Objekte voneinander entfernt sind bzw. wie weit sie sich durchdringen. Gelegentlich fragt man nach der minimalen Verschiebung, um die Objekte wieder zu trennen, ob die Entfernung zwischen den Objekten eine vorgegebene Toleranzschwelle unterschreitet (Toleranzprüfung), oder, bei gegebenen Bewegungen, wann die nächste Kollision eintritt (ETA, engl. estimated time of arrival). Bei einem wichtigen Spezialfall bewegt sich ein Teil der Objekte nicht (statische Szene). Sind mehr als zwei Objekte gleichzeitig zueinander auf Kollision zu prüfen, spricht man von einem n-Körper-Problem.[1]

Die wichtigste Anforderung an Algorithmen zur Kollisionserkennung ist die Laufzeit. Die meisten Ansätze zur Lösung des allgemeinen Problems der Kollisionserkennung repräsentieren Objekte durch ihre Oberflächen; diese werden durch geometrische Primitive wie Polygonnetze, Splines oder algebraische Oberflächen (Menge aller Wurzeln eines Polynoms $f : \mathbb{R}^3 \to \mathbb{R}$) beschrieben. Für nicht oberflächenrepräsentierte Objekte wie z. B. durch dreidimensionale Primitive (Kugeln, Kapseln,

[1] In Anlehnung an das klassische Problem aus der Physik, bei dem die Lösungen für die Bewegungsgleichungen von mehreren durch Gravitation interagierenden Körpern gesucht sind.

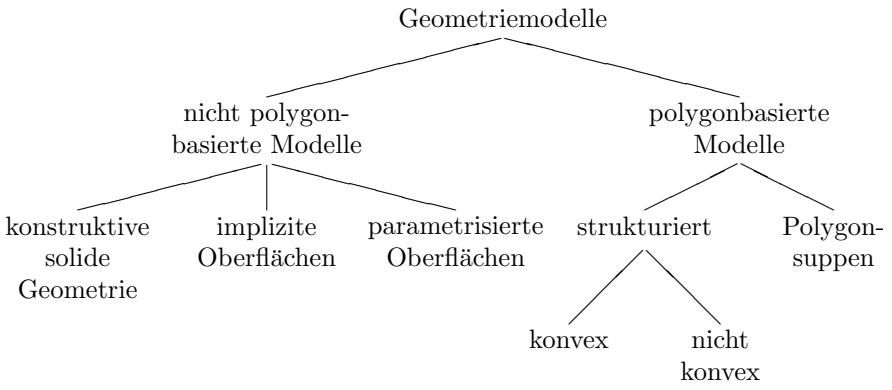


Abbildung 2.1: Repräsentationsformen geometrischer Objekte nach Lin und Gottschalk [1]

Lozenges, Zylinder, Ellipsoide, achsenausgerichtete Quader, orientierte Quader usw.) beschriebene Objekte oder volumetrische Objekte existieren ebenfalls Methoden zur Kollisionserkennung. Abbildung 2.1 gibt einen Überblick über Repräsentationsformen geometrischer Objekte.

2.1.2 Einordnung des Schweißzellenproblems

Für unsere Zwecke ist die Geometrie der Roboter, des Werkstücks und der Hindernisse zu modellieren. Da es sich um zweckorientiert entworfene, industriell gefertigte Maschinen und Bauteile handelt,[2] dominieren einfache, starre, nicht kompliziert geschwungene geometrische Formen (zu Kabeln u. ä. siehe die Abschnitte 1.3.4 und 3.1.10). Für deren Modellierung eignen sich sowohl die konstruktive solide Geometrie als auch eine strukturierte, polygonbasierte Beschreibung der Oberflächen. Bei letzterer lassen sich nicht konvexe Bauteile aus konvexen Bauteilen zusammensetzen. Beide Repräsentationen erlauben eine einfache und intuitive Modellierung der Roboter, des Werkstücks und der Hindernisse.

Nur die Roboter bewegen sich, alle anderen Objekte (Werkstück und Hindernisse) sind statisch. Keines der Objekte wird deformiert; die Bewegungen der Roboter sind durch Maximalgeschwindigkeiten und -beschleunigungen beschränkt. Es handelt sich um ein n-Körper-Problem im Sinne des vorherigen Abschnitts.

2.1.3 Methoden

Um Rechenzeit einzusparen sind eine Vielzahl verschiedener Techniken vorgeschlagen worden, u. a. Hüllvolumen-Bäume, geometrisches Schließen, Partitionierungen des Raums sowie algebraische und analytische Methoden. Manche Algorithmen nutzen die zeitliche Kohärenz sich bewegender Objekte aus, d. h. sie profitieren davon, wenn die Kollisionsabfragen in zeitlichen Intervallen vorgenommen werden, in denen sich die Positionen der Objekte nur wenig verändern. Für einen vollständigeren Überblick über Methoden zur Kollisionserkennung verweisen wir auf die Literatur [1, 4]; im Folgenden beschreiben wir kurz einige Ansätze:

Hüllvolumen-Bäume [9, 5, 10, 6, 11, 12] Die exakte Geometrie der Objekte wird durch einfachere geometrische Primitive eingehüllt. Tests auf Kollision finden zuerst zwischen diesen Hüllvolumen (engl. bounding volumes) statt; kollidieren diese, werden die genaueren Geometriedaten verwendet. Die Algorithmen unterscheiden sich hauptsächlich in der Wahl der Hüllvolumen und der verwendeten hierarchischen Struktur. Als Hüllvolumen werden häufig achsenausgerichtete Quader

[2] Dies gilt nicht notwendigerweise für das Werkstück; in unserem Fall handelt es sich aber typischerweise um Teile des Karosserierohbaus, die durch die oben genannten Techniken modelliert werden können.

(AABB, engl. Axis Aligned Bounding Boxes), orientierte Quader oder Kugeln verwendet, da diese schnelle Kollisionstests erlauben. Das Konzept kann iterativ angewendet werden (Hierarchie von zunehmend gröberen Hüllvolumen); es können mehrere Arten von Hüllvolumina verwendet werden. Datenstrukturen zur Organisation der Hüllvolumen sind meist Bäume.

Auf Hüllvolumen basierende Methoden sind schnell, wenn die Objekte weit auseinander liegen, aber langsamer, wenn die Objekte nahe beieinander sind oder es viele Kontaktstellen gibt. Bei der Wahl der Hüllvolumina ist zwischen der Genauigkeit der Approximation und der Aufwändigkeit der Kollisionstests abzuwägen; z. B. erlauben Kugeln sehr schnelle Kollisionstests, können die Objektgeometrie aber meist nur bedingt genau approximieren. Bewegen oder verändern sich die Objekte, müssen die Hüllvolumen-Bäume dynamisch aktualisiert werden.

Lin-Canny-Algorithmus [16, 18] Der Lin-Canny-Algorithmus ist auf wiederholte Kollisionsabfragen für zwei sich bewegende konvexe Polyeder bei zeitlicher Kohärenz ausgelegt. Nach einer Vorverarbeitung der Objekte zu Beginn der Abfragen erfordern diese in der Praxis im Schnitt konstante Zeit. Der Algorithmus basiert auf Voronoi-Regionen und merkt sich diejenigen Merkmale der Polyeder, die sich räumlich am nächsten sind. Die Kenntnis der nächsten Merkmale erlaubt eine einfache Berechnung der Distanz.

Der Voronoi-Clip-Algorithmus [17] funktioniert ähnlich wie der Lin-Canny-Algorithmus, kann aber auch sich schneidende Polyeder (und dadurch auch manche nicht konvexen Polyeder) behandeln.

Gjk-Algorithmus [70, 15] Dieser Algorithmus nach Gilbert, Johnson und Keerthi berechnet, wie der Lin-Canny-Algorithmus auch, die Distanz zwischen zwei konvexen Polyedern. Es handelt sich im Wesentlichen um ein iteratives Gradientenabstiegsverfahren zur Approximation der Minkowski-Summe[3] unter Verwendung des Simplex-Algorithmus. Dabei wird eine Folge von Simplizes erzeugt, deren Minimalpunkt gegen den Minimalpunkt des Distanzpolyeders konvergiert.

Für den GJK-Algorithmus wie auch für den Lin-Canny-Algorithmus existieren eine Reihe von Verbesserungen des ursprünglichen Verfahrens.

Aufteilung des Raums [2] Bei diesem Ansatz nutzt man die räumliche Kohärenz der Objekte aus und unterteilt den Raum in Teilräume; jedes Objekt befindet sich in einem oder mehreren dieser Teilräume. Es sind nur diejenigen Objekte auf Kollision zu testen, die sich in gleichen Teilräumen befinden. Die verschiedenen Verfahren unterscheiden sich vor allem in der Art wie sie den Raum aufteilen.

Bekannte Verfahren sind Octrees, BSP-Bäume (engl. Binary Space Partitioning) und die uniforme Aufteilung des Raums. Octrees und BSP-Bäume teilen den Raum adaptiv auf, d. h. Bereiche mit vielen Objekten werden feiner unterteilt (mehr Teilräume) als Gegenden mit wenig Objekten. Dabei wird bei Octrees der Raum in acht gleich große Teilräume aufgeteilt; diese werden gegebenenfalls wieder in jeweils acht Teilräume aufgeteilt usw. Bei BSP-Bäumen wird der Raum durch eine (frei wählbare) Ebene geteilt; die so entstandenen zwei Teilräume werden bei Bedarf jeweils wieder durch eine Ebene geteilt usw. Die uniforme Aufteilung des Raums benötigt viel Speicherplatz und ist besonders für Objekte gleicher Größe geeignet.

[3] Unter der Minkowski-Summe zweier Objekte $A, B \subseteq \mathbb{R}^3$ versteht man die Menge $A \oplus B := \{a + b \mid a \in A, b \in B\}$. Die Minkowski-Summe zweier konvexer Polyeder ist wieder ein konvexer Polyeder. Der *Distanzpolyeder* zweier konvexer Polyeder A und B ist $A \oplus (-B)$. Die beiden Polyeder A und B kollidieren genau dann, wenn der Koordinatenursprung innerhalb des Distanzpolyeders liegt; die Distanz zwischen A und B entspricht genau der Distanz des Distanzpolyeders vom Ursprung. Den Punkt des Distanzpolyeders mit dem geringsten Abstand zum Ursprung nennt man den Minimalpunkt des Distanzpolyeders. [70]

Name	Sprache	Plattform & Compiler	Version
FreeSolid	C, C++	Windows (VC++ 5.0), Posix (g++ 2.8.1)	0.1 vom 14.11.2002
Opcode	C++	Windows[4] (VC++ 6.0)	1.3 vom 5.6.2003
QuickCd	C	Unix (Linux, Irix, Solaris) (gcc)	1.00
Rapid	C++	keine Angabe	2.01 vom 6.9.1997
Swift++	C++	Windows (VC++ 6.0), Unix	1.1
V-Clip	C++	keine Angabe	keine Angabe
V-Colllide	C++	Unix (Irix, HPUX, Solaris) (g++ 2.7)	1.1 vom 1.2.1998
verb. Gjk	C	Posix (g++ 3.3[5])	2.4 vom Juli 1998

Tabelle 2.1: Programmbibliotheken zur Kollisionserkennung, Stand Januar 2004

2.1.4 Implementierungen

Wir stellen eine Auswahl an Programmbibliotheken zur Kollisionserkennung vor; Tabelle 2.1 gibt einen Überblick.

FreeSolid [15] (`www.win.tue.nl/~gino/solid/`) Auf einer verbesserten Version des GJK-Algorithmus basierende Programmbibliothek zur Kollisionserkennung, einschließlich der Berechnung der Überschneidungstiefe. Die Ausnutzung zeitlicher Kohärenz wird unterstützt. Objekte können aus Primitiven (Quadern, Kegeln, Zylindern, Kugeln) sowie konvexen Polygonen und Polyedern zusammengesetzt sein. Für konvexe Polyeder kann das (ebenfalls frei verfügbare) Qhull Programmpaket[6] für konvexe Hüllen benutzt werden. Die Bibliothek ist unter der GNU LGPL (engl. Library General Public License) verfügbar.

Opcode [8] (`www.codercorner.com/Opcode.htm`) Opcode (Optimized Collision Detection) ist eine Programmbibliothek zur speichereffizienten Kollisionserkennung, einschließlich der Bestimmung des Kollisionsortes. Die Objekte können als „Polygonsuppe“ (unstrukturierte Menge von Polygonen) vorliegen. Basiert auf einer speicherplatzoptimierten hierarchischen Repräsentation der Objekte mit achsenausgerichteten Quadern (AABB, Axis Aligned Bounding Boxes) als Hüllvolumen. Zeitliche Kohärenz wird unterstützt. Die Bibliothek ist frei verfügbar.

QuickCd [9, 10] (`www.ams.sunysb.edu/~jklosow/quickcd/`) QuickCd (engl. Quick Collision Detection) ist eine Programmbibliothek zur exakten Kollisionserkennung auf der Basis von Hüllvolumen-Bäumen. Die Objekte können als „Polygonsuppe“ (unstrukturierte Menge von Polygonen, hier Dreiecke) vorliegen. Als Hüllvolumen werden k-dops (engl. discrete orientation polytopes) verwendet. Dabei handelt es sich um konvexe Polyeder, deren Seiten Normalen aus einer festen (kleinen) Menge von k Vektoren haben. Die Bibliothek ist für nicht kommerzielle Zwecke frei verfügbar.

Rapid [5, 6] (`www.cs.unc.edu/~geom/OBB/OBBT.html`) Rapid (engl. Robust and Accurate Polygon Interference Detection) ist eine kleine, einfach zu benutzende Programmbibliothek zur Kollisionserkennung für zwei Objekte, einschließlich der Bestimmung des Kollisionsorts. Die Objekte können als „Polygonsuppe“ (unstrukturierte Menge von Polygonen, hier Dreiecke) vorliegen. Zeitliche Kohärenz und n-Körper-Verarbeitung werden nicht unterstützt. Der verwendete Algorithmus basiert auf einer hierarchischen Repräsentation der Objekte mit orientierten Quadern

[4] Version 1.2 wurde auf Unix (gcc 3.2) portiert, `www.paassen.tmfweb.nl`.
[5] Angabe ermittelt vom Autor dieser Arbeit, nicht durch offizielle Quellen zur Programmbibliothek.
[6] `www.thesa.com/software/qhull`.

(OBB, engl. Oriented Bounding Box) als Hüllvolumen und einem Separationsachsen-Theorem zur schnellen Kollisionsüberprüfung bei OBBs. Die Bibliothek ist für nicht kommerzielle Zwecke frei verfügbar.

Das PQP (engl. Proximity Query Package) [11, 12] (`http://www.cs.unc.edu/~geom/SSV/`) Programmpaket ist eine Variante von RAPID, die auch Distanzberechnungen und Toleranzprüfung unterstützt. Als Hüllvolumen werden über Quadern „geschwenkte“ Kugeln verwendet. PQP ist für nicht kommerzielle Zwecke frei verfügbar.

SWIFT++ [18] (`www.cs.unc.edu/~geom/SWIFT/`) SWIFT++ (engl. Speedy Walking via Improved Feature Testing for Non-Convex Objects) bietet Kollisionstests einschließlich der Bestimmung der Kontaktflächen, Toleranzprüfung sowie approximativer und exakter Entfernungsbestimmung. Zeitliche Kohärenz wird unterstützt. Das QHULL Programmpaket[7] für konvexe Hüllen wird vorausgesetzt. Die Bibliothek ist für nicht kommerzielle Zwecke frei verfügbar.

SWIFT++ ist eine Erweiterung der SWIFT Programmbibliothek; im Gegensatz zu dieser können auch nicht konvexe Objekte verarbeitet werden. SWIFT selbst ist eine Implementierung eines verbesserten Lin-Canny-Algorithmus.

V-CLIP [17] (`www.merl.com/projects/vclip/`) Die V-CLIP (engl. Voronoi-Clip) Programmbibliothek beruht auf einer Variante des Lin-Canny-Algorithmus und bietet Kollisionserkennung und Abstandsberechnung für konvexe sowie aus konvexen Objekten zusammengesetzte Objekte. Für konvexe Dekompositionen kann die QHULL Programmbibliothek[7] verwendet werden. Die Bibliothek ist für nicht kommerzielle Zwecke frei verfügbar.

V-COLLIDE [7] (`www.cs.unc.edu/~geom/V\_COLLIDE/`) Das V-COLLIDE (engl. Accelerated Collision Detection for VRML) Programmpaket setzt auf der RAPID Bibliothek auf, die zur eigentlichen Kollisionsüberprüfung eingesetzt wird. Zusätzlich ist ein n-Körper-Test enthalten, der unter Berücksichtigung zeitlicher Kohärenz bestimmt, welche Objekte überprüft werden müssen. Die Objekte können als „Polygonsuppe“ (unstrukturierte Menge von Polygonen, hier Dreiecke) vorliegen und zur Laufzeit entfernt und hinzugefügt werden. Die Bibliothek ist für nicht kommerzielle Zwecke frei verfügbar.

VERBESSERTER GJK [14] (`web.comlab.ox.ac.uk/oucl/work/stephen.cameron/distances`) Auf einer Verbesserung des GJK-Algorithmus basierende Programmbibliothek zum Berechnen des Abstands zweier konvexer Polyeder bei Ausnutzung zeitlicher Kohärenz. Die Bibliothek ist für nicht kommerzielle Zwecke frei verfügbar.

Die Programmbibliothek I-COLLIDE wurde von V-COLLIDE und RAPID bzw. SWIFT++ abgelöst. IMMPACT (`www.cs.unc.edu/~geom/MMC/`) ist eine Programmbibliothek zur Kollisionserkennung bei sehr großen Modellen (Anzahl der Dreiecke im zweistelligen Millionenbereich).

2.2 Wegplanung und Ansteuerung

Für einen vollständigeren Überblick über das Gebiet der Wegplanung verweisen wir auf entsprechende Übersichtsartikel [20, 19] in der Literatur. Auf die (vom verwendeten Robotertyp abhängige) Programmierung der Roboter gehen wir nicht näher ein. Für andere Themen der Ansteuerung interessieren wir uns ausschließlich im Rahmen der Wegplanung.

[7] `www.thesa.com/software/qhull`.

2.2.1 Gegenstand des Gebiets

Beim allgemeinen Problem der Wegplanung (auch Pfadplanung, engl. motion planning, path planning) geht es in der einfachsten Variante darum, einen kollisionsfreien Weg für ein sich bewegendes Objekt zwischen bekannten statischen Hindernissen zu finden. Fortgeschrittene Fragestellungen beschäftigen sich mit dynamischen Umgebungen, kinematischen Restriktionen des sich bewegenden Roboters, Unsicherheiten in Kontrolle und Sensorik, der Einbeziehung physikalischer Faktoren wie Reibung, der Koordination mehrerer sich bewegender Roboter sowie der Berechnung optimaler Trajektorien.

Anwendungen für Algorithmen zur Wegplanung finden sich in der Robotik (insbesondere bei mobilen autonomen Robotern), der (virtuellen) Prüfung von Maschinen und Anlagen auf Wartbarkeit bzw. Zugänglichkeit, der computergestützten Chirurgie, der Animation im Bereich Computergrafik, dem Entwurf von Medikamenten („drug docking“) sowie der Faltung von Proteinen.

Kollisionserkennung ist — explizit oder implizit — Bestandteil praktisch aller Algorithmen zur Wegplanung (zur Kollisionserkennung siehe Abschnitt 2.1).

2.2.2 Einordnung des Schweißzellenproblems

Der Wegplanungsaspekt des Schweißzellenproblems beinhaltet die Koordination mehrerer Roboter mit kinematischen Restriktionen in einer statischen Umgebung; dabei sind zeitlich optimale, kollisionsfreie Wege gesucht.

Reibung spielt praktisch keine Rolle, da es keine Berührungen zwischen den sich bewegenden Objekten gibt. Unsicherheiten und Ungenauigkeiten in der Kontrolle der auszuführenden Bewegungen werden ausschließlich im Rahmen der Ansteuerung berücksichtigt.

Wegplanung und Optimierung sind im Schweißzellenproblem eng miteinander verbunden; dies wird insbesondere in der kontinuierlichen Formulierung (Definition 1.5) deutlich. Das Schweißzellenproblem lässt sich sowohl als Wegplanungsproblem als auch als Optimierungsproblem auffassen.

Die Berechnung optimaler Trajektorien ist für Fragen der Ansteuerbarkeit von Bedeutung.

2.2.3 Komplexität

Ein *vollständiger Pfadplaner* (auch exakter Pfadplaner) ist ein Algorithmus zur Wegplanung, der einen kollisionsfreien Weg findet, falls ein solcher existiert, und ansonsten feststellt, dass es keinen solchen Weg gibt. John Reif hat 1979 bewiesen [24], dass das vollständige Pfadplanungsproblem PSPACE-hart ist, wenn die Anzahl der Freiheitsgrade des Roboters Teil der Eingabe ist. Der Beweis kodiert die Konfiguration einer polynomiell platzbeschränkten Turingmaschine in den Freiheitsgraden des Roboters und benutzt die Hindernisse, um die Berechnung der Maschine zu simulieren. Der Roboter bestand dabei aus einer Folge von durch Gelenke verbundenen dreidimensionalen Polyedern; die Hindernisse wurden ebenfalls als Polyeder modelliert.

Ist die Anzahl der Freiheitsgrade fest, also nicht Teil der Eingabe, liegt das Problem in P. Erweiterungen des Problems erhöhen in der Regel die Komplexität erheblich; so ist das vollständige Wegplanungsproblem bei sich bewegenden Hindernissen sogar für eine feste Anzahl an Freiheitsgraden der Roboter PSPACE-hart. [19]

Möglichkeiten zum Umgang mit dieser Komplexität sind der Entwurf von Approximationsschemata, probabilistischen Algorithmen bzw. Heuristiken für das allgemeine Wegplanungsproblem oder die Beschränkung auf Probleme mit sehr kleinem Freiheitsgrad (zwei bis drei) der Roboter. Die letzte Möglichkeit ist für das Schweißzellenproblem nicht relevant, da die verwendeten Industriero”boter meist fünf bis sechs Gelenke haben.

2.2.4 Methoden

Viele Algorithmen beruhen auf dem Begriff des Konfigurationsraums (siehe Abschnitt 1.4.1). Ist der (freie) Konfigurationsraum diskret, hat er eine explizite oder implizite Repräsentation als Graph. Dabei entsprechen die Knoten den (freien) Konfigurationen und die Kanten den möglichen Aktionen des Roboters (eine Aktion überführt eine Konfiguration des Roboters in eine andere Konfiguration; welche Aktionen verfügbar sind, ist durch die vom Robotertyp abhängige Ansteuerung bestimmt). Die Kanten können mit Kosten, z. B. der Dauer einer Aktion oder der im Raum zurückgelegten Entfernung, behaftet sein. Die Suche nach einem (kürzesten) Weg von einer Startkonfiguration zu einer Zielkonfiguration entspricht dann der Suche nach einem (kürzesten) Weg von einem Startknoten zu einem Zielknoten in diesem Graphen.

A^*-Algorithmus [21] Der A^*-Algorithmus war einer der ersten Pfadplanungsalgorithmen. Er setzt eine Repräsentation des freien Konfigurationsraums als Graph (Sichtbarkeitsgraph) und eine untere Schranke für die Kosten eines kürzesten Weges von jedem Knoten zum Zielknoten voraus. Es handelt sich um eine Variante von Dijkstras Algorithmus für kürzeste Wege [80], bei der die noch zu untersuchenden Knoten nicht nach den bisherigen Kosten für den Weg vom Startknoten zu diesen Knoten sortiert sind, sondern nach der Summe von den bisherigen Kosten zu dem Knoten und der unteren Schranke für die weiteren Kosten zum Zielknoten. Der Algorithmus spart dadurch Besuche von Knoten ein.

Zell-Dekomposition [19] Bei diesem vollständigen Pfadplanungsvefahren wird der freie Konfigurationsraum rekursiv mit Hilfe eines algebraischen Theorems in zylindrische Zellen aufgeteilt. Diese bilden die Knoten eines Graphen; benachbarte Zellen werden durch Kanten verbunden. Ein kollisionsfreier Weg kann dann innerhalb des Graphen gesucht werden. Die Laufzeit für die Dekomposition ist doppelt exponentiell, mit der Anzahl der Freiheitsgrade des Roboters im äußeren Exponenten.

Stochastische Strassenkarte [23] Beim PRM-Verfahren (engl. Probabilistic Road Map) werden zufällig Stichproben aus dem Konfigurationsraum gezogen; diejenigen Stichproben, die im freien Konfigurationsraum liegen, werden durch lokale Pfade miteinander verbunden. Unter gewissen Annahmen[8] verringert sich die Wahrscheinlichkeit, keinen Pfad zu finden, obwohl ein solcher existiert, exponentiell mit der Anzahl der Stichproben. PRM-Verfahren sind numerisch robust und einfach zu implementieren. Es existieren verschiedene Anpassungen an nicht holonomische (kinematische) und dynamische Nebenbedingungen, verschiedene Optimierungskriterien sowie dynamische Umgebungen. Die Verteilung der Stichproben kann an die Umgebung angepasst werden, z. B. indem mehr Stichproben aus Bereichen der Umgebung gezogen werden, in denen bisher nur wenige Konfigurationen durch lokale Pfade miteinander verbunden werden konnten.

Ob eine Stichprobe im freien Konfigurationsraum liegt, kann durch Algorithmen zur Kollisionserkennung festgestellt werden. PRM-Implementierungen verwenden einen Großteil ihrer Rechenzeit (90% oder mehr [20]) für Kollisionserkennung und Distanzberechnungen; ihre Verwendung setzt daher schnelle Algorithmen zur Kollisionserkennung voraus.

[8] Zur Analyse des PRM-Algorithmus existieren zwei Ansätze: Abdeckung des Raums und Pfadisolierung.

Beim Ansatz über die Abdeckung des Raums wird die Anzahl der benötigten Stichproben zu Eigenschaften des Konfigurationsraums in Beziehung gesetzt. Im Wesentlichen darf es keine „engen Passagen“ im freien Konfigurationsraum geben und jede freie Konfiguration des Roboters muss durch den lokalen Pfadplaner mit einer Menge von freien Konfigurationen verbunden werden können, deren Volumen mindestens einen ϵ Bruchteil vom Gesamtvolumen aller freien Konfigurationen beträgt, für ein festes $\epsilon \in [0, 1[$.

Ausschlaggebend sind dabei die engen Passagen, da diese eine große Anzahl an Stichproben erfordern; solche engen Passagen können nicht nur durch die Hindernisse bedingt sein, sondern auch durch das Zusammenspiel von Hindernissen und der (eingeschränkten) Bewegungsfähigkeit des Roboters. Ansätze zum Umgang mit engen Passagen sind die Anpassung der Stichprobenverteilung an die Umgebung sowie die Berücksichtigung von Stichproben mit geringer Überschneidungstiefe im nicht freien Konfigurationsraum.

Beim Pfadisolierungsansatz wird die Anzahl benötigter Stichproben zu den Eigenschaften eines existierenden Pfades in Beziehung gesetzt, im Wesentlichen Länge und Distanzfunktion des Pfades.

Potentialfeld [19] Bei diesem physikalisch motivierten Verfahren induzieren die Hindernisse sowie der Start- und der Endpunkt des gesuchten Weges ein Feld, wobei die Hindernisse abstoßend, Start- und Endpunkt dagegen anziehend wirken. Der Roboter wird dann entlang des Feldgradienten bewegt. Ein Problem dieses Ansatzes ist, dass der Roboter in lokalen Minima des Feldes stecken bleiben kann. Die Potentialfeld Heuristik kann als lokaler Wegplaner verwendet werden, indem Start- und Endpunkt nahe beeinander gewählt werden (um die Wahrscheinlichkeit des Steckenbleibens in einem lokalen Minimum zu verringern).

Randomisierte Wegplaner [25] (RMP, engl. Randomized Motion Planner) erhält man aus der Potentialfeld Methode durch Hinzunahme zufälliger Bewegungen, um aus lokalen Minima entkommen zu können.

Die vorgestellten Algorithmen lassen sich auch für die Koordination mehrerer Roboter verwenden. Dazu fasst man alle Roboter zusammen als einen großen, räumlich nicht zusammenhängenden Roboter auf, indem man das kartesische Produkt über die Konfigurationsräume der einzelnen Roboter bildet. Die Anzahl der Freiheitsgrade des so konstruierten Roboters ist die Summe der Freiheitsgrade der ursprünglichen Roboter. Man beachte, dass der freie Konfigurationsraum des zusammengesetzten Roboters eine Teilmenge des Produkts der freien Konfigurationsräume der ursprünglichen Roboter ist, wobei diejenigen Konfigurationen wegfallen, bei denen Roboter kollidieren.

Eine dem Problem der Wegplanung verwandte Aufgabenstellung ist die Ablaufplanung bei der Montage (engl. assembly sequencing, assembly planning); dabei geht es darum, für die einzelnen Teile eines Produkts eine geeignete Reihenfolge und entsprechende Bewegungen der Roboter zu finden, um sie zum fertigen Produkt zusammenzubauen. Hierfür existiert ein implementiertes Verfahren, welches auf einem Abhängigkeitsgraph (NDBG, engl. Non-Directional Blocking Graph) beruht. Das Verfahren ist jedoch anfällig für Rundungsfehler und seine Komplexität wächst exponentiell mit der Dimension des Raums der zulässigen Bewegungen. [20] Für Spezialfälle existieren effizientere Verfahren. [19]

Zur Berechnung optimaler Trajektorien im Rahmen der Ansteuerung gibt es Algorithmen; so stellen etwa Sonja Macfarlane und Elizabeth Croft [27] einen effizienten Algorithmus auf der Basis von Polynomen fünften Grades vor, der für einen gegebenen Weg (LSPB, engl. Linear Segments with Parabolic Bounds) nahezu zeitoptimale, erschütterungsbeschränkte Bahnen berechnet. Auch für die Anpassung von Wegen an die Beschränkungen der Steuereinheiten gibt es Algorithmen. [28] Diese führen sogar aufgrund besserer Ansteuerbarkeit und Nachführbarkeit zu schneller ausführbaren Bahnen.

2.3 Optimierung

Auf dem Gebiet der Optimierung (vormals vor allem bei betriebswirtschaftlichem Bezug auch als Operations Research bezeichnet) beschäftigt man sich mit den Extrema (Minima und Maxima) einer Funktion. Zahlreiche Anwendungen finden sich sowohl in der Wirtschaft (z. B. in den Bereichen Organisation, Logistik, Produktion und Telekommunikation) als auch im wissenschaftlichen Bereich (z. B. in Mathematik und Informatik sowie im chemischen und biomedizinischen Bereich).

Zum Thema Optimierung existiert ein umfangreiches Angebot an Literatur und anderweitig verfügbarer Information. [32, 35, 33, 65, 34]

Wir beschäftigen uns im Folgenden nicht mit Bedingungen für die Existenz oder die Eindeutigkeit von Extrema u. ä., sondern ausschließlich mit Methoden zu ihrer Bestimmung.

2.3.1 Gegenstand des Gebiets

Sei $\mathcal{S}$ eine Menge. Unter *Optimierung* verstehen wir die Aufgabe, zu einer gegebenen Funktion $f : \mathcal{S} \to \mathbb{R}$ ein $x \in \mathcal{S}$ zu finden, so dass $f(x)$ minimal[9] ist, d. h. $\forall y \in \mathcal{S} : f(x) \leq f(y)$. Dabei gehen wir davon aus, dass f und $\mathcal{S}$ so beschaffen sind, dass ein Minimum existiert.[10]

Unter einem *Optimierungsproblem* versteht man eine parametrisierte Menge von Optimierungsaufgaben; die Elemente dieser Menge heißen *Instanzen* des Optimierungsproblems.

Beispiel 2.1 Ein aus der Schulzeit bekanntes, sehr einfaches Optimierungsproblem ist: „Gegeben ein Polynom zweiten Grades, bestimme seine Extrema in $\mathbb{R}$." Eine Instanz dieses Optimierungsproblems ist „finde die Extrema von $f(x) = \frac{1}{5}x^2 + 3x + \frac{1}{2}$". □

Man unterscheidet verschiedene Arten von Optimierungsproblemen:

Global vs. lokal Bei einem globalen Optimierungsproblem bezieht sich die Optimalitätsbedingung auf den gesamten Suchraum, bei einem lokalen Optimierungsproblem dagegen nur auf Umgebungen der lokalen Optima.

Kontinuierlich vs. diskret Im kontinuierlichen Fall ist $\mathcal{S}$ überabzählbar unendlich, im diskreten Fall ist $\mathcal{S}$ abzählbar unendlich oder endlich.[11] Letzteren Fall bezeichnet man auch als kombinatorische Optimierung.

Uneingeschränkt vs. Nebenbedingungen Im uneingeschränkten Fall wird f auf ganz $\mathcal{S}$ minimiert, wobei $\mathcal{S}$ üblicherweise explizit spezifiziert wird. Im eingeschränkten Fall wird f nur auf einer, meist durch Nebenbedingungen implizit spezifizierten, Teilmenge von $\mathcal{S}$ minimiert.

Einzelkriterium vs. Mehrfachkriterien Bei Mehrfachkriterien sind statt einer Funktion f mehrere Funktionen $f_1, \ldots, f_n$ zu minimieren. Diese stimmen in der Regel in ihren Minima nicht überein, so dass man stattdessen nach Pareto optimalen (auch nicht dominierten) Argumenten sucht. Dabei heißt ein Punkt $x \in \mathcal{S}$ *Pareto optimal*, wenn es keinen anderen Punkt $x' \in \mathcal{S}$ gibt, so dass $f_i(x') \leq f_i(x)$ für $1 \leq i \leq n$, wobei mindestens eine der Ungleichungen echt sein muss.[12]

Deterministisch vs. stochastisch Im stochastischen Fall enthält die Zielfunktion einen zufälligen Anteil. Üblicherweise ist der Erwartungswert von f zu minimieren.

Wir bezeichnen die Menge $\mathcal{S}$ als *Suchraum* und f als *Zielfunktion*; Lösungen können anschaulich als Punkte im Suchraum aufgefasst werden. Oft ordnet man jeder Lösung eine *Nachbarschaft* aus nahebei liegenden Lösungen zu, im diskreten Fall z. B. durch Anwenden von lokalen Operationen. Die Wahl von $\mathcal{S}$ ist nicht immer offensichtlich; oft ist es zum Entwurf von Algorithmen nützlich, $\mathcal{S}$ größer zu wählen als die Menge der für die gegebene Aufgabenstellung zulässigen Lösungen, etwa als die Menge aller „syntaktisch" korrekten Eingaben.

2.3.2 Einordnung des Schweißzellenproblems

Fasst man das Schweißzellenproblem als Optimierungsproblem auf, so handelt es sich um ein globales deterministisches Optimierungsproblem mit Nebenbedingungen. Optimiert wird nach einem einzelnen Kriterium, der Gesamtbearbeitungszeit. Ob der Suchraum kontinuierlich oder diskret ist, hängt von der Menge $\mathcal{M}$ der zulässigen Bewegungen ab.

[9] Die Suche nach einem Maximum von f ist äquivalent zur Suche nach einem Minimum von $-f$.

[10] Für $\mathcal{S} := \mathbb{R} \setminus \{0\}$ und $f(x) := x^2$ geht das einzige Minimum durch eine Einschränkung des Suchraums verloren.

[11] Die genaue Definition lautet: Eine Teilmenge X eines größeren topologischen Raums Y heißt *diskret* (in Y), falls zu jedem Punkt $x \in X$ eine Umgebung $U \subseteq Y$ existiert, so dass $X \cap U = \{x\}$.

[12] Formal: $\nexists x' \in \mathcal{S}, x' \neq x : \ \forall i \in \{1, \ldots, n\} : f_i(x') \leq f_i(x) \ \wedge \ \exists i \in \{1, \ldots, n\} : f_i(x') < f_i(x)$.

2.3.3 Methoden zur globalen Optimierung

Wir unterscheiden deterministische und stochastische[13] Verfahren sowie exakte Methoden, Approximationsalgorithmen und Heuristiken. Exakte Methoden liefern stets ein globales Optimum, haben jedoch oft eine exponentielle worst-case Laufzeit.[14] Approximationsalgorithmen berechnen Lösungen, deren Qualität in einem garantierten Verhältnis zum Optimum steht, z. B. Abweichung um höchstens einen konstanten Faktor; erlaubt der Algorithmus die Vorgabe einer solchen Schranke für die Qualität der Lösung als Teil der Eingabe, spricht man von einem Approximationsschema [86]. Approximationsalgorithmen stellen in der Regel einen Kompromiss (engl. trade off) zwischen Laufzeit und Güte der Lösung dar. Heuristiken sind Algorithmen, die keine Aussage über die Qualität der gefundenen Lösung treffen. Es handelt sich meist um Verfahren, die erfahrungsgemäß auf der Mehrzahl der in der Praxis auftretenden Fälle schnell und mit hinreichend gutem Ergebnis arbeiten. Sie werden in der Regel empirisch analysiert und verglichen.

Für unsere Art von Optimierungsproblem gibt es u. a. die folgenden Methoden:

2.3.3.1 Deterministische Methoden

Deterministische Verfahren sind, obwohl dies oft der Fall ist, nicht notwendigerweise exakt. Für viele Probleme existieren maßgeschneiderte Algorithmen; wir beschränken uns an dieser Stelle auf allgemein anwendbare deterministische Methoden.

Dynamisches Programmieren [80] Ursprünglich das Finden optimaler Entscheidungen in einem (zeitlich) sequentiellen Entscheidungsprozess gemäß Richard Bellmanns Optimalitätsprinzip,[15] nach dem in jeder Phase des Entscheidungsprozesses die verbleibendenden Entscheidungen optimal in Bezug auf den aktuellen Zustand (determiniert durch die bisherigen Entscheidungen) sein müssen. Man versteht unter dynamischem Programmieren eine Strategie zum Entwurf von Algorithmen, bei der die Lösung des Problems auf die Lösung kleinerer Instanzen des Problems zurückgeführt wird, üblicherweise indem man eine Variable ihre möglichen Werte annehmen lässt und jeweils die optimale Lösung für das verbleibende Problem berechnet. Die Lösungen der kleineren Probleminstanzen werden gespeichert, um unnötige Mehrfachberechnungen zu vermeiden; üblicherweise geschieht dies in einer Tabelle, die entweder rekursiv oder, bei den kleinsten Probleminstanzen beginnend, iterativ berechnet wird.

Branch & Bound [37, 32] Verzweige & Begrenze-Algorithmen basieren auf der Idee, den Suchraum systematisch zu durchsuchen und dabei Teile auszulassen, welche die optimale Lösung nicht enthalten können. Voraussetzung dafür ist, dass man den Suchraum in kleinere Probleminstanzen aufteilen und untere Schranken für die beste in einem solchen Teil enthaltene Lösung berechnen kann.

Man beginnt mit der Erstellung eines Baums, dessen Wurzel mit dem Suchraum beschriftet ist. Dieser wird partitioniert[16] und jedes Kind der Wurzel wird mit einer der Partitionen beschriftet.

[13] Man beachte den Unterschied zwischen stochastischen Optimierungsproblemen und stochastischen Optimierungsmethoden.

[14] Um stets ein globales Optimum zu finden, muss ein Algorithmus entweder die Zielfunktion auf dem ganzen Suchraum auswerten oder problemspezifische Eigenschaften der Zielfunktion ausnutzen. Man denke zur Verdeutlichung an Funktionen wie

$$f : \mathbb{R} \to \mathbb{R}, \quad f(x) := \begin{cases} 0 & \text{falls } x = 0, \\ 1 & \text{sonst.} \end{cases}$$

Viele angewandte Probleme, darunter auch das Schweißzellenproblem, haben einen sehr großen Suchraum.

[15] *„An optimal policy has the property that whatever the initial state and the initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions.“* Für die Namensgebung „Dynamisches Programmieren“ waren politische Gründe ausschlaggebend.

[16] Eine Aufteilung in echte Teilmengen, deren Vereinigung den Suchraum ergibt, ist ausreichend, erhöht aber den Rechenaufwand.

Das Verfahren wird rekursiv fortgesetzt (z. B. über Tiefensuche). Enthält eine Partition nur eine Lösung oder stimmen obere und untere Schranke der Partition überein, wird diese mit der bisher besten Lösung verglichen und gegebenenfalls übernommen. Knoten, deren untere Schranke größer als die bisher beste Lösung ist, werden nicht expandiert. Oft startet man mit einer durch eine Heuristik ermittelten guten Lösung als untere Schranke.

Lineare Programmierung [39, 41] Streng genommen ist die lineare Programmierung (LP, engl. Linear Programming) eine Form der Modellierung und kein Algorithmus. Ein lineares Programm (in Standardform) ist ein Optimierungsproblem der Form „gegeben eine $m \times n$ Matrix A über $\mathbb{R}$ sowie $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, finde $x \in \mathbb{R}^n$ so dass $c^T x$ minimal ist und $Ax = b$ sowie $x \geq 0$ gilt.“ Es handelt sich also um eine spezielle Klasse von Optimierungsproblemen mit zu minimierender linearer Funktion $f : \mathbb{R}^n \to \mathbb{R}$, $f(x) = c^T x$ und linearen Nebenbedingungen. Müssen alle oder einige der Komponenten von x ganzzahlig sein, so spricht man von reiner bzw. gemischter ganzzahliger Programmierung (ILP bzw. IP, engl. Integer (Linear) Programming, MIP bzw. MILP, engl. Mixed Integer (Linear) Programming). Ein weiterer Spezialfall ist die 0-1 ILP mit $x \in \{0,1\}^n$.

Kombinatorische Optimierungsprobleme können in der Regel als lineare Programme formuliert werden. In der Praxis wird die lineare Programmierung häufig eingesetzt, da es gute Implementierungen von Algorithmen zur Lösung von linearen Programmen gibt.

Nichtlineare Programmierung [40] Eine der linearen Programmierung ähnliche Form der Modellierung für nichtlineare Funktionen mit nichtlinearen Nebenbedingungen: Gegeben eine Funktion $F : \mathbb{R}^n \to \mathbb{R}$ sowie l Funktionen $g_1, \ldots, g_l : \mathbb{R}^n \to \mathbb{R}$ und m Funktionen $h_1, \ldots, h_m : \mathbb{R}^n \to \mathbb{R}$, finde $x \in \mathbb{R}^n$ so dass $F(x)$ minimal ist und $g_1(x) = \ldots = g_l(x) = 0$ sowie $h_1(x) \geq 0, \ldots, h_m(x) \geq 0$ gilt. Spezialfälle sind lineare Nebenbedingungen, lineare Nebenbedingungen mit quadratischer Zielfunktion (QP, engl. Quadratic Programming), Optimierung ohne Nebenbedingungen und die lineare Programmierung.

Intervall-Methoden [81, 33] Durch Intervall-Arithmetik[17] werden globale Informationen wie obere und untere Schranken, Lipschitz-Konstanten und höhere Ableitungen für die Einschränkung der Zielfunktion auf (rechteckigen) Gebieten berechnet. Diese können dann z. B. für Verzweige & Begrenze-Algorithmen verwendet werden.

Für Intervall-Methoden muss die Zielfunktion als algebraischer Ausdruck vorliegen. Die Berechnung von Ableitungsinformationen ist nicht zwingend notwendig, erhöht aber die Effizienz der Algorithmen.

Fortsetzungs- und Glättungsmethoden [42] Fortsetzungsmethoden (engl. continuation methods) beruhen auf der Idee, aus dem ursprünglichen Optimierungsproblem ein vereinfachtes Optimierungsproblem zu konstruieren, dieses zu lösen und anschließend schrittweise wieder zur ursprünglichen Version überzugehen (fortzusetzen). Der Übergang erfolgt meist durch einen zusätzlichen Parameter $\lambda \in [0,1] \subset \mathbb{R}$, wobei man für $\lambda = 0$ die vereinfachte Zielfunktion, für $\lambda = 1$ die ursprüngliche Zielfunktion (oder umgekehrt) und für $0 < \lambda < 1$ Übergangsformen der Zielfunktionen erhält. Lässt man λ schrittweise von 0 gegen 1 gehen, so kann man in jedem Schritt eine Lösung auf Basis der vorherigen Lösung finden (die Bahn durch diese Lösungen wird als „Nullkurve“ bezeichnet). Zur parametrisierten Vereinfachung der Zielfunktion kommen verschiedene Transformationen in Frage, z. B. die logarithmische Barriere-Funktion [42] oder die Konvolution mit der Dichtefunktion der Normalverteilung [43].

Fortsetzungs- und Glättungsmethoden sind keine exakten Methoden.

[17] In der Intervall-Arithmetik wird mit (abgeschlossenen) Intervallen anstatt mit reellen Zahlen gerechnet. Durch Rundung „nach außen“ können echte obere und untere Schranken für die Ergebnisse von Fließkommaberechnungen garantiert werden. [81]

2.3.3.2 Stochastische Methoden

Stochastische Algorithmen zeichnen sich durch die Verwendung von Zufallsbits aus. Die in diesem Abschnitt vorgestellten Algorithmen sind nicht exakt.

Multistart und Clustering [34] Multistart ist eine der ältesten Methoden. Sie basiert darauf, einen lokalen Optimierungsalgorithmus von mehreren (zufällig gewählten) Punkten im Suchraum aus zu starten. Um das mehrfache Finden desselben Minimums zu vermeiden, können Clustering Methoden verwendet werden. Dabei werden die Punkte zu Clustern („Haufen“) zusammengefasst, von denen man annimmt, dass sie zum selben lokalen Minimum gehören. Der lokale Optimierungsalgorithmus wird dann nur noch von einem Punkt eines Clusters aus ausgeführt.

Die Multistart-Suche versagt bei Problemen mit sehr vielen lokalen Minima. Clustering Methoden sind für uneingeschränkte Optimierungsprobleme entwickelt worden und eher für niedrigdimensionale Probleme geeignet; da viele Punktproben nötig sind, um die Cluster zu identifizieren, sollte die Zielfunktion günstig auszuwerten sein.

Kontrollierte Zufällige Suche [66] Bei der Methode der kontrollierten zufälligen Suche (CRS, engl. Controlled Random Search) wird eine anfängliche Menge von Punkten iterativ kontrahiert, indem der schlechteste (größte) Punkt durch einen besseren Punkt ersetzt wird. Dieser wird durch ein globales Verfahren (z. B. durch Bildung eines Simplex aus einem Teil der Punkte mit anschließender Spiegelung eines der Punkte des Simplex am Zentrum der restlichen Punkte) oder ein lokales Verfahren (z. B. durch zufällige Wahl um den besten Punkt gemäß einer β-Verteilung) bestimmt.

Das Verfahren eignet sich für die Optimierung sowohl mit als auch ohne Nebenbedingungen; es wurde ursprünglich für nicht differenzierbare Zielfunktionen entwickelt.

Metropolis-Algorithmus [44] Bei diesem von Metropolis et al. entwickelten Verfahren wird, ausgehend von einem zufälligen Startpunkt, eine Folge von Punkten im Suchraum konstruiert, wobei die Wahl des nächsten Punktes zufällig ist und nur vom vorherigen Punkt abhängt (Markov Eigenschaft). Dabei werden bessere Punkte immer akzeptiert, schlechtere dagegen nur mit einer von der Größe der Verschlechterung abhängigen Wahrscheinlichkeit. Die Verteilung dieser Wahrscheinlichkeit ändert sich im Verlauf des Algorithmus nicht. Die bisher beste gefundene Lösung wird separat mitgeführt.

Simuliertes Abkühlen [46] Das simulierte Abkühlen (engl. Simulated Annealing) geht auf eine Analogie zur Physik[18] zurück und ist eine Weiterentwicklung des Metropolis-Algorithmus, bei der sich die Akzeptanzwahrscheinlichkeiten für Verschlechterungen mit der Zeit verringern. Die Veränderung kann nichtadaptiv (nur von der Zeit abhängig) oder adaptiv (abhängig auch vom Verlauf des Algorithmus) sein.

Das Verfahren wurde für die kombinatorische Optimierung entworfen.

Baum Abkühlen [45] Das Baum Abkühlen (engl. tree annealing) ist eine auf Intervallen basierende Variante des simulierten Abkühlens, die für kontinuierliche Optimierungsprobleme entwickelt wurde. Sie basiert auf einer Repräsentation des Suchraums durch k-d Bäume, wobei jeder Knoten ein Intervall in dem ihm zugeordneten Freiheitsgrad (gesteuert über die Tiefe des Knotens) halbiert. Jeder Knoten entspricht dadurch einem Hyperquader. Der Baum wächst adaptiv und stochastisch im Verlauf des Algorithmus, wobei diejenigen Bereiche ausgebaut (genauer untersucht) werden, in denen niedrige Funktionswerte liegen.

Das Verfahren eignet sich für Probleme mit wenigen Freiheitsgraden (nicht mehr als 30).

[18] Erhitzt man einen Festkörper, z. B. ein Metall, bis er schmilzt, und lässt ihn dann langsam abkühlen, so ordnen sich die Teilchen des Körpers von alleine in einer regelmäßigen Gitterstruktur an; dies entspricht einem Zustand niedrigster Energie. Dabei muss man um so länger warten, je näher das Material an seinen Gefrierpunkt kommt. Geschieht das Abkühlen zu schnell, entstehen Defektstellen in der Struktur, an denen das Material spröde und nicht belastbar ist. Solche Stellen können durch zwischenzeitliches Erhöhen der Temperatur wieder aufgelöst werden.

Evolutionäre Algorithmen [48] Ein Oberbegriff für Algorithmen, die auf Prinzipien der natürlichen biologischen Evolution beruhen. Charakteristisch ist die Verwendung mehrerer Lösungen (Population von Individuen); diese werden durch Reproduktion/Rekombination und Mutation variiert. Ein Selektionsmechanismus sorgt für die Auswahl von Individuen anhand ihrer Fitness (Wert der Zielfunktion).

Historisch[19] unterscheidet man genetische Algorithmen, Evolutionsstrategien, evolutionäre Programmierung und genetische Programmierung. Man kann den Metropolis-Algorithmus und seine Abarten als Spezialfälle evolutionärer Algorithmen betrachten. Eine Stärke evolutionärer Algorithmen ist ihre inhärente Parallelität.

Scatter Search [67] Die Streuungssuche ist eine Variante der evolutionären Algorithmen, deren zentrales Element eine Referenzmenge ist, die einen Teil der gefundenen Lösungen enthält (Kriterien für die Aufnahme sind Optimalität und Diversität). Aus diesen werden durch Rekombination neue Lösungen erzeugt, die sowohl innerhalb als auch außerhalb der konvexen Hülle der Referenzmenge liegen; anschließend werden diese durch lokale Suchverfahren verbessert und ein Teil wird in die Referenzmenge übernommen. Das Verfahren wird fortgesetzt bis sich die Referenzmenge nicht mehr ändert. Sind weitere Iterationen gewünscht, kann die Referenzmenge durch teilweise Neuinitialisierung wieder diversifiziert werden.

Ameisen-Algorithmen [68] Zum Gebiet der Schwarmintelligenz[20] gehörende Algorithmen, die sich am Vorbild natürlicher Ameisen orientieren. Diese kommunizieren über Pheromone (Duftstoffe), die sie auf von ihnen benutzten Wegen deponieren; dabei werden bei Entscheidungen Wege mit mehr Pheromonen bevorzugt. Zur Optimierung verwendet man „künstliche“ Ameisen, die Lösungen sukzessive aufbauen; dabei werden Elemente entsprechend ihrer Pheromonmarkierung bevorzugt. Hat eine Ameise eine Lösung aufgebaut, werden die verwendeten Elemente entsprechend der Güte der Lösung mit Pheromonen markiert. Die Pheromone bauen sich mit der Zeit ab.

Statistische Methoden [69] Bei diesem Bayes-Ansatz[21] wird angenommen, dass die Zielfunktion durch eine (zu wählende) parametrisierte stochastische Funktion modelliert werden kann. Die Parameter dieser Funktion werden jeweils auf Basis aller bisher gezogenen Stichproben geschätzt, während die Wahl der nächsten Stichprobe aufgrund des aktuellen Modells nach statistischen Gesichtspunkten erfolgt. Dabei ist abzuwägen zwischen gut untersuchten Teilen des Suchraums, in denen bereits gute Lösungen gefunden wurden und bisher wenig untersuchten Teilen oder Teilen mit sehr unterschiedlichen Funktionswerten.

Eine Schwierigkeit dieser Herangehensweise liegt in der richtigen Wahl der Modellierungsfunktion. Die meisten statistischen Ansätze sind für kontinuierliche Optimierungsprobleme entworfen worden. Statistische Methoden lohnen sich erst, wenn die Auswertung der Zielfunktion teuer genug ist, um den Aufwand für die statistische Analyse zur Wahl der nächsten Stichprobe zu rechtfertigen.

[19] Klassischerweise arbeiteten genetische Algorithmen mit binären Zeichenketten fester Länge zur Repräsentation der Lösung, während Evolutionsstrategien zur Parameteroptimierung auf kontinuierlichen Suchräumen dienten. Evolutionäre Programmierung befasste sich mit dem Erlernen deterministischer endlicher Automaten; bei der genetischen Programmierung bestand der Suchraum aus Programmen. Die Unterscheidung scheint nicht mehr zeitgemäß, da heutige evolutionäre Algorithmen oft Mischformen sind und problemspezifische Repräsentationen und Operatoren verwenden.

[20] Unter Schwarmintelligenz versteht man die Eigenschaft eines aus einfachen, mit ihrer Umgebung lokal interagierenden Agenten aufgebauten Systems, kohärente funktionale globale Verhaltensmuster zu zeigen. Insbesondere verfügen die betrachteten Systeme über keine zentrale Kontrolle und kein (explizites) globales Modell der Umwelt. Man spricht auch von Partikelschwärmen oder Multi-Agenten-Systemen.

[21] Der Unterschied zwischen bayesianischer und frequentistischer Auffassung liegt in der Rolle des Parameters eines statistischen Modells: Nach der frequentistischen Auffassung sind nur die Beobachtungen Zufallsgrößen, nicht jedoch der Parameter. Bei der bayesianischen Sichtweise wird auch der Parameter als Zufallsgröße betrachtet, für die a priori eine Verteilung angenommen wird; diese wird anschließend auf die Beobachtungen bedingt. Der bayesianische Ansatz erlaubt das Einbringen von Vorwissen.

2.3.3.3 Hybride Verfahren und Metaheuristiken

Hybride Verfahren sind Mischformen verschiedener Optimierungsmethoden. Metaheuristiken sind allgemeine Verfahren zum Einsatz spezialisierter (problemspezifischer) Heuristiken.

Tabu-Suche [57] Eine Metaheuristik, die auf folgender Idee basiert: Viele iterative Suchverfahren ordnen jeder Lösung einen Teil des Suchraums als Nachbarschaft zu; die in einem Iterationsschritt neu hinzukommende Lösung enstammt dann der Nachbarschaft der aktuell betrachteten Lösung. Bei der Tabu-Suche wird diese Nachbarschaft anhand der bereits gefundenen Lösungen eingeschränkt (Teile der Nachbarschaft sind tabu), um die wiederholte Erkundung von Teilen des Suchraums zu vermeiden. Tabu-Suche wird hauptsächlich für kombinatorische Optimierung verwendet.

Memetische Algorithmen [60, 59] Auch hybride genetische Algorithmen, genetische lokale Suche. Eine populationsbasierte Kombination aus lokalen Suchverfahren und genetischen Algorithmen. Dabei werden etwa durch Initialisierung, Rekombination oder Mutation neu entstandene Individuen einem lokalen Suchverfahren, z. B. Gradientenabstieg, unterworfen. Degeneriert die Population (feststellbar z. B. durch ein Diversitätsmaß wie die Shannon-Entropie), so behält man einen Teil der Lösungen und initialisiert den Rest der Population neu. Memetische Algorithmen werden hauptsächlich in der kombinatorischen Optimierung angewandt.

Iterierte lokale Suche [61] Eine auch als iterierter Abstieg, Markovketten mit großer Schrittweite und iterierte Lin-Kernighan-Methode bekannte Metaheuristik. Beginnend mit einer Startlösung wird iterativ eine Sequenz von Lösungen erzeugt. Dazu wird die aktuelle Lösung perturbiert und anschließend einem lokalen Optimierungsverfahren, z. B. Gradientenabstieg oder lokaler Suche, unterworfen. Genügt die resultierende Lösung einem Akzeptanzkriterium, wird sie zur neuen aktuellen Lösung. Das Verfahren wird iterativ fortgesetzt.

Grasp [63] Die Grasp-Metaheuristik (engl. Greedy Randomized Adaptive Search Procedure), eine Variante der Multistart-Methode, basiert auf iterierter lokaler Optimierung. Die Startlösung für jede Iteration wird sukzessive mittels einer „gierigen“ Bewertungsfunktion neu konstruiert: Alle für den verbleibenden Teil der Lösung in Frage kommenden Elemente werden bewertet; aufgrund der Bewertungen wird ein Element stochastisch ausgewählt und zur Lösung hinzugefügt; anschließend wird wieder neu bewertet usw. Es existieren zahlreiche Erweiterungen des Verfahrens, z. B. Einbeziehung der Ergebnisse vorheriger Iterationen in die Konstruktionsphase und lokale Suche auf Teillösungen innerhalb einer Iteration.

2.3.4 Verwandte Probleme

Wir stellen einige Probleme aus dem Bereich der kombinatorischen Optimierung vor, die einen Bezug zum Schweißzellenproblem haben. Für die in Kapitel 3 erfolgende Modellierung des Schweißzellenproblems als kombinatorisches Optimierungsproblem sind insbesondere die Probleme TSP, CVRP und JSP von Interesse.

2.3.4.1 Das Problem des Handlungsreisenden

Eines der bekanntesten Probleme der kombinatorischen Optimierung ist das Problem des Handlungsreisenden, bei dem es um das Finden einer kürzesten Rundreise durch einen vollständigen Graphen geht:

Definition 2.1 (TSP, engl. Traveling Salesman Problem) Gegeben ein vollständiger kantengewichteter Graph, finde einen Hamiltonkreis (einen geschlossenen Weg, der jeden Knoten genau einmal enthält) mit minimalem Gewicht. □

Bei ungerichteten Graphen spricht man vom STSP (engl. Symmetric TSP), bei gerichteten Graphen vom ATSP (engl. Asymmetric TSP). Das STSP ist ein Spezialfall des ATSP. Umgekehrt lässt sich ein ATSP als STSP darstellen, indem man die Anzahl der Knoten verdoppelt.[22] Die Vollständigkeit des Graphen ist ebenfalls keine echte Einschränkung: Man weist jeder im unvollständigen Graphen nicht vorhandenen Kante ein so großes Gewicht zu, dass sie nicht in der Lösung enthalten sein kann.[23] Die folgenden Varianten [72] können ebenfalls als TSP formuliert werden (meist durch Veränderung der Kantengewichte):

Flaschenhals-TSP (engl. bottleneck TSP) Statt den Gesamtkosten der Rundreise wird das größte enthaltene Kantengewicht minimiert. Formulierung als TSP durch exponentiell große Kantengewichte.

TSP mit Mehrfachbesuchen (TSPM, engl. TSP with Multiple Visits). Die Knoten dürfen öfter als einmal im Weg enthalten sein. Formulierung als TSP durch Kosten kürzester Wege als Kantengewichte.

Botenproblem (engl. messenger problem) An Stelle eines Hamiltonkreises wird ein Hamiltonpfad gesucht; Anfangs- und Endknoten sind gegeben. Formulierung als TSP durch Zuweisen eines sehr großen negativen Gewichts an die Kante zwischen Anfangs- und Endknoten.

Minimaler Hamiltonpfad Gesucht ist ein Hamiltonpfad mit minimalem Gewicht; im Unterschied zum Botenproblem sind Anfangs- und Endknoten nicht vorgegeben. Formulierung als TSP durch Hinzufügen eines neuen Knotens, dessen Kanten mit einem sehr großen negativen Gewicht gewichtet sind.

Haufen-TSP (engl. Cluster TSP) Der Graph ist partitioniert und die Rundreise unterliegt der zusätzlichen Bedingung, dass die Knoten einer Partition nacheinander besucht werden müssen. Formulierung als TSP durch Addition eines großen Gewichts zu allen Kanten zwischen Partitionen.

Verallgemeinertes TSP (GTSP, engl. Generalized TSP) Der Graph wird partitioniert; gesucht ist ein Weg mit minimalem Gewicht, der genau einen Knoten aus jeder Partition enthält. Das TSP ergibt sich als Spezialfall mit Partitionsgröße 1. Formulierung als TSP durch Erzwingen einer festen Reihenfolge innerhalb einer Partition und Vertauschen der ausgehenden Kantengewichte von Eintritts- und Austrittsknoten.[24]

TSP mit m Handlungsreisenden Sei V die Menge der Knoten des Graphen und $s \in V$ der Startknoten. Gesucht sind eine Partitionierung von $V \setminus \{s\}$ in m Mengen $V_1, \ldots, V_m \subset V$ sowie m Hamiltonkreise in $V_1 \cup \{s\}, \ldots, V_m \cup \{s\}$ mit minimalem Gesamtgewicht. Formulierung als TSP durch $m - 1$ weitere Startknoten mit gleichen Kantengewichten wie der ursprüngliche Startknoten.[25]

[22] Die Idee ist, die Richtung der Kanten durch Paare von Knoten nachzubilden. Sei (V, E) der gerichtete Graph. Der ungerichtete Graph (V', E') enthält dann für jeden Knoten $x \in V$ einen weiteren Knoten $x' \in V$. Die Gewichte der gerichteten Kanten $(x, y) \in E$ gibt man den Kanten $(x, y') \in E'$. Die Kanten zwischen x und x' erhalten ein sehr großes negatives Gewicht (um sicherzustellen, dass ein minimaler Weg sie enthält), alle anderen Kanten erhalten ein sehr großes positives Gewicht (um sicherzustellen, dass ein minimaler Weg sie nicht enthält). [75]

[23] Ein solches Gewicht kann z. B. die Summe M aller Kantengewichte im ursprünglichen Graphen sein. Ist der Wert der Lösung größer als M, hat das TSP keine Lösung im unvollständigen Graphen.

[24] O. B. d. A. sei der Graph gerichtet. Man ordnet die Knoten einer Partition ringförmig an, indem man Kanten zwischen aufeinanderfolgenden Knoten ein sehr großes negatives Gewicht gibt. Damit legt ein erster besuchter Knoten der Partition auch immer den letzten besuchten Knoten fest; diesem weist man jeweils die aus der Partition herausführenden Kantengewichte des zuerst besuchten Knoten zu. Dadurch kann ein Hamiltonkreis im modifizierten Graphen als Rundreise mit einem Knoten pro Partition im ursprünglichen Graphen interpretiert werden.

[25] Im gerichteten Fall ersetzt man den Startknoten durch m Knotenpaare, wobei der erste Knoten die ausgehenden, der zweite Knoten die eingehenden Kanten des Startknotens erhält. Der zweite Knoten jedes Knotenpaars ist mit dem ersten Knoten eines anderen Knotenpaars durch eine gerichtete Kante verbunden.

Stellvertretend[26] für kantenorientierte Routenplanungsprobleme (engl. arc routing problems) [77], bei denen die Anforderungen nicht aus zu besuchenden Knoten, sondern aus zu besuchenden Kanten bestehen, stellen wir folgendes Problem vor:

Gemischtes windiges ländliches Postbotenproblem[27] (MWRPP, engl. Mixed Windy Rural Postman Problem) Gegeben sind ein gemischter Graph mit gerichteten und ungerichteten Kanten sowie eine Teilmenge der gerichteten und eine Teilmenge der ungerichteten Kanten. Gesucht ist ein geschlossener Weg mit minimalem Gewicht, der alle Kanten der beiden Teilmengen enthält. Formulierung als GTSP auf einem neuen Graphen, dessen Knoten den Kanten im ursprünglichen Graphen entsprechen und kürzesten Weglängen als Kantengewichten. [74]

In der Entscheidungsvariante[28] ist das Problem des Handlungsreisenden NP-vollständig. Ist $P \neq NP$, so gibt es keinen polynomiellen δ-Approximationsalgorithmus mit $\delta > 1$ für das TSP. Appendix B im Buch von Gregory Gutin und Abraham Punnen [72] gibt einen Überblick über weitere Komplexitätsergebnisse für das TSP und seine Varianten.

Zwei andere Verallgemeinerungen des TSP sind das TSP mit Reihenfolge-Nebenbedingungen (manche Knoten müssen vor anderen Knoten besucht werden) und

Zeitabhängiges TSP Für jede Kante des Graphen $G = (V, E)$ sind $|V|$ Gewichte gegeben. Die Kosten für die Kante vom i-ten zum $(i+1)$-ten Knoten einer Tour sind durch das i-te Gewicht dieser Kante gegeben. Das TSP ergibt sich als Spezialfall mit identischen Gewichten für jede Kante.

Das TSP hat viele Anwendungen in zahlreichen Gebieten gefunden, u. a. auch in der Ablaufsteuerung und -planung (inklusive der Bestückung von Leiterplatten) sowie der zellenbasierten Fertigung.

2.3.4.2 Routenplanungsprobleme für Fahrzeuge

Unter VRPs (engl. Vehicle Routing Problems) versteht man eine dem TSP verwandte Klasse von Problemen, bei denen es um die Routenplanung für eine Flotte von Lieferfahrzeugen geht. Die zu besuchenden Orte (Kunden) werden durch die Knoten eines Graphen modelliert, die Entfernungen (oder Fahrzeiten) zwischen diesen durch Kantengewichte. Jedem Ort ist eine Nachfrage zugeordnet. Die Fahrzeuge starten von einem speziellen Knoten, dem Depot, aus und kehren zu diesem zurück. Gesucht sind Touren für die Fahrzeuge, die gewissen Nebenbedingungen wie Kapazitätsbeschränkungen, Zeitfenstern oder Terminen genügen und deren Gesamtlänge minimal ist.

Definition 2.2 (CVRP, engl. Capacitated Vehicle Routing Problem) Gegeben ein vollständiger, kantengewichteter Graph, bei dem jedem Knoten eine Nachfrage zugeordnet ist, ein als Depot ausgezeichneter Knoten sowie eine Fahrzeugkapazität. Gesucht ist eine Menge von im Depot startenden Touren, deren Gesamtlänge minimal ist; dabei darf für jede Tour die Summe der Nachfragen der besuchten Knoten die Fahrzeugkapazität nicht überschreiten und jeder Knoten

[26] Beispielsweise sind das gemischte chinesische Postboten-Problem (MCPP, engl. Mixed Chinese Postman Problem), das windige chinesische Postboten-Problem, das windige ländliche Postboten-Problem (WRPP, engl. Windy Rural Postman Problem) und das Stapelkran-Problem (SCP, engl. Stacker Crane Problem) Spezialfälle des MWRPP.

[27] Das Wort „windig“ entstammt einer bildlichen Motivation für asymmetrische Kantengewichte: Je nachdem, ob der Postmann Rückenwind oder Gegenwind hat, benötigt er für die gleiche Strecke unterschiedlich lange.

[28] Bei Entscheidungsproblemen (auch Wortproblemen) wird nach der Zugehörigkeit eines Wortes zu einer formalen Sprache (einer gegebenen Menge von Wörtern) gefragt.

In der Entscheidungsvariante lautet das TSP „Gegeben ein kantengewichteter Graph G und eine natürliche Zahl $k \in \mathbb{N}_0$, gibt es eine Rundreise auf G mit Kosten höchstens k?“ Die dazugehörige formale Sprache ist $\{(G, k) \mid G \text{ ist kantengewichteter Graph und } G \text{ besitzt eine Rundreise mit Kosten} \leq k \in \mathbb{N}_0\}$.

außer dem Depot muss insgesamt genau einmal besucht werden. Die Anzahl der Touren (Fahrzeuge) ist nicht vorgegeben. □

Unter der Gesamtlänge der Touren ist hierbei die Summe der Kosten aller Touren zu verstehen. Gelegentlich ist die Anzahl der Fahrzeuge vorgegeben, manchmal darf die Länge einer Tour eine vorgegebene Schranke nicht überschreiten. Das CVRP ist eine Kombination aus dem TSP und dem NP-harten Bin Packing-Problem[29].

2.3.4.3 Ablaufsteuerung und -planung

Das Gebiet der Ablaufsteuerung und -planung (engl. sequencing and scheduling) umfasst Fragestellungen zur optimalen Verteilung von Ressourcen unter Nebenbedingungen, insbesondere in den Bereichen der Produktionsplanung, des Projektmanagements und der Steuerung von Computern. Beispiele sind die Aufteilung von herzustellenden Produkten auf Fertigungsmaschinen, das Verteilen von Aufgaben auf Mitarbeiter und das Abarbeiten von Programmen auf einem Prozessor. Es existiert eine Vielzahl von Problemstellungen, von denen wir stellvertretend zwei vorstellen:

Definition 2.3 (SOP, engl. Sequential Ordering Problem) Gegeben ein gerichteter kantengewichteter Graph, Anfangs- und Endknoten sowie eine Menge an Reihenfolge-Nebenbedingungen (der Form „Knoten i muss vor Knoten j bearbeitet werden“). Finde einen Hamiltonpfad vom Anfangs- zum Endknoten, der die Nebenbedingungen erfüllt. □

Das SOP ist eine Variante[30] des ATSP mit Nebenbedingungen. Varianten des SOP sind disjunktive Reihenfolge-Nebenbedingungen (der Form „x Knoten aus den Knoten $y_1, y_2, y_3, \ldots$ müssen vor Knoten z besucht werden“) und weiche Reihenfolge Nebenbedingungen, bei denen die Verletzung einer Nebenbedingung möglich, aber mit Kosten verbunden ist. [38]

Definition 2.4 (JSP, engl. Job Shop Problem) Gegeben sind eine endliche Menge von Maschinen und eine endliche Menge von Aufgaben, jede bestehend aus einer festgelegten Reihenfolge von Operationen. Jeder Operation ist eine Zeitdauer zugeordnet und eine Maschine, auf der sie abgearbeitet werden kann. Eine Maschine kann immer nur eine Operation gleichzeitig ausführen; die Ausführung einer Operation kann nicht unterbrochen werden. Finde eine Reihenfolge für die Abarbeitung der Operationen auf den Maschinen, welche die Gesamtbearbeitungszeit minimiert. □

Es gibt zahlreiche Varianten des JSP. Als Entscheidungsproblem ist es NP-hart; es bleibt auch dann NP-hart, wenn man es auf drei Maschinen und gleiche Bearbeitungszeiten einschränkt, ebenso bei Einschränkung auf drei Aufgaben oder unterbrechbare Operationen. [64] Exakte Lösungen der schwierigeren Probleme im Bereich Ablaufsteuerung und -planung sind bisher nur für kleine Probleminstanzen möglich.

2.4 Weiterführendes

In allen besprochenen Gebieten — Kollisionserkennung, Wegplanung, Ansteuerung und Optimierung — existieren zahlreiche Algorithmen für unterschiedliche Problemstellungen, die sich jedoch deutlich in ihrer Anwendbarkeit auf das Schweißzellenproblem unterscheiden.

[29] Gegeben sind n Objekte mit Gewichten in (0,1). Gesucht ist eine Aufteilung der Objekte auf die minimale Anzahl an Behältern der Kapazität 1.

[30] Die Unvollständigkeit des Graphen und die Angabe eines Start- und Endknotens sind keine echten Einschränkungen, siehe die Transformationen im Unterabschnitt zum TSP auf Seite 28.

Die Kollisionserkennung lässt sich gut von den anderen Aspekten des Schweißzellenproblems trennen und als eigenständiges Problem betrachten. In Anbetracht der in Abschnitt 2.1.4 vorgestellten Programmpakete zur schnellen Kollisionserkennung setzen wir in den folgenden Kapiteln Algorithmen zur Kollisionserkennung, Abstandsberechnung usw. als vorhanden voraus.

Der Optimierungs- und der Wegplanungsaspekt des Schweißzellenproblems sind eng miteinander verbunden. Die Komplexität der in der Literatur untersuchten Probleme legt nahe, dass die Kosten optimaler Algorithmen für das Schweißzellenproblem prohibitiv wären. Im nächsten Kapitel werden daher Vereinfachungen des Schweißzellenproblems vorgestellt, die sowohl die Komplexität des Problems reduzieren als auch eine bessere Trennung von Optimierungs- und Wegplanungsaspekt erlauben.

Die in Abschnitt 2.3.4 vorgestellten kombinatorischen Optimierungsprobleme sind eng miteinander verwandt. Teilweise lassen sie sich sogar ineinander überführen; so geben Beck, Prosser und Selensky [78] eine Transformation vom CVRP mit Zeitfenstern zum offenen JSP und umgekehrt. Dabei bringen für das CVRP entworfene Verfahren schlechte Ergebnisse für transformierte Instanzen des offenen JSP und umgekehrt. Ursächlich hierfür scheint die „untypische“ Struktur der transformierten Probleminstanzen zu sein.

Mit Ausnahme[31] des TSP sind die meisten der beschriebenen kombinatorischen Optimierungsprobleme für größere Instanzen derzeit in der Praxis nicht exakt lösbar. In den meisten Fällen existieren aber zahlreiche problemspezifische, meist probabilistische Heuristiken, die in der Praxis schnell gute Ergebnisse liefern, ebenso wie einige der in Abschnitt 2.3.3 vorgestellten nicht exakten Methoden.

[31] David Applegate, Robert Bixby, Vašek Chvátal und William Cook [73] haben 2001 ein TSP mit 15 112 Knoten exakt gelöst. Die Rechenzeit betrug 22,6 Jahre, zurückgerechnet auf einen Compaq EV6 Alpha Prozessor mit 500 MHz.

Kapitel 3

Modellierung

Die in Kapitel 1 mit den Definitionen 1.5 und 1.6 vorgestellten vollständigen Formulierungen des Schweißzellenproblems geben dieses zwar genau wieder, sind dadurch aber aufwändig und enthalten sehr viele Freiheitsgrade, was den Suchraum stark vergrößert. Die Komplexität der in Kapitel 2 vorgestellten, deutlich einfacheren Probleme legt ebenfalls eine Vereinfachung der vollständigen Problemformulierung nahe. Hinzu kommt, dass die Definitionen 1.5 und 1.6, um sich nicht auf ein bestimmtes Ansteuerungsmodell festzulegen, über die Menge $\mathcal{M}$ der zulässigen Bewegungen parametrisiert sind. Um zu besser handhabbaren Problemen zu kommen, vereinfachen wir im Folgenden die ursprünglichen Definitionen auf Basis konkreter Ansteuerungsmodelle und erhalten so mehrere, sich in Grad und Art der Vereinfachung unterscheidende Varianten des Schweißzellenproblems.

3.1 Vereinfachungsmöglichkeiten

Wir untersuchen zuerst systematisch, welche Möglichkeiten wir zur Vereinfachung haben. Als Ansatzpunkte bieten sich sowohl die Forderungen in den Definitionen 1.5 und 1.6 als auch die in den Abschnitten 1.2 und 1.3 aufgezählten Erweiterungen und schwierigen Aspekte an. Während die Forderung nach der Vollständigkeit der Lösung (alle Schweißpunkte müssen bearbeitet werden) in unserem Szenario[1] keine Angriffsfläche bietet, lassen sich bei allen anderen Punkten Vereinfachungen vornehmen.

3.1.1 Optimalität

In beiden Formulierungen wird die Optimalität der Lösung gefordert. In Anbetracht der Komplexität des Schweißzellenproblems erscheint dies unrealistisch. In unserem Anwendungsszenario kann bereits eine Verbesserung der Gesamtbearbeitungszeit die Produktionsgeschwindigkeit steigern oder, in der Planungsphase, die Bearbeitung von mehr Schweißpunkten bzw. die Einsparung von Robotern ermöglichen. Eine weitere Einsatzmöglichkeit für schnell berechenbare gute Lösungen ist die (interaktive) Unterstützung der Offline-Programmierer der Roboter. Ziel eines Algorithmus sollte es also sein, in einer akzeptablen Anzahl von Rechenschritten eine möglichst gute Lösung zu finden.

[1] Betrachtet man nicht eine einzelne Roboterstation, sondern eine aus solchen Stationen aufgebaute Fertigungsstraße, kann man die Forderung nach Vollständigkeit der Lösung aufweichen und nicht bearbeite Schweißpunkte zu vorherigen oder späteren Stationen verschieben.

3.1.2 Kollisionsfreiheit

Kollisionsfreiheit ist beim Einsatz auf echten Roboterstationen unverzichtbar. Dennoch können nicht kollisionsfreie Lösungen nützlich sein, z. B. kann eine kollisionsarme Lösung als Vorlage für einen menschlichen Planer dienen.

Da die Wegplanung eng mit der Kollisionserkennung zusammenhängt — Kollisionen sind der einzige Grund, um von einer zeitlich optimalen Trajektorie zwischen zwei Schweißpunkten abzuweichen — entfällt mit einem Verzicht auf Kollisionsfreiheit auch ein großer Teil der Wegplanung.

Es gibt zwei Arten der Vereinfachung bezüglich der Kollisionsfreiheit: Den völligen Verzicht auf Kollisionsfreiheit (entspricht dem Weglassen von Punkt 3 in den Definitionen 1.5 und 1.6) und die Forderung nach kollisionsarmen Lösungen. Bei der zweiten Möglichkeit handelt es sich um eine weiche Nebenbedingung, die ein Maß für den Umfang der auftretenden Kollisionen erfordert. Ein solches Maß kann z. B. das Integral des Überschneidungsvolumens aller Roboter über die Gesamtbearbeitungszeit sein (Kollisionsintegral). [88]

3.1.3 Nebenbedingungen

Ähnlich wie bei der Kollisionsfreiheit ist die Einhaltung mancher Nebenbedingungen unverzichtbar; hat ein Roboter z. B. nicht die passende Schweißzange für einen Schweißpunkt, ist es ihm unmöglich, diesen zu bearbeiten. Auch hier kann eine Lösung, welche die Nebenbedingungen verletzt, nützlich sein, etwa wenn die Lösung auf einfache Weise manuell korrigiert werden kann.

Wieder bietet sich als Vereinfachung sowohl der Verzicht auf Nebenbedingungen als auch die Forderung nach möglichst wenig Verletzungen der Nebenbedingungen an. Der Übergang zu weichen Nebenbedingungen erfordert ein Maß für den Umfang der Verletzungen der Nebenbedingungen. Denkbar ist auch eine Mischung aus harten und weichen Nebenbedingungen.

3.1.4 Sicherheitsabstand

Die Forderung nach einem Sicherheitsabstand lässt sich vollständig im Rahmen der Kollisionserkennung behandeln. Da sich deren Rolle in den Definitionen 1.5 und 1.6 auf die Verwendung der Abstandsmaße $d_{i,j}$ und d_i beschränkt, kann die explizite Forderung nach einem Sicherheitsabstand O. B. d. A. entfallen.

Einige der in Abschnitt 2.1.4 vorgestellten Programmbibliotheken unterstützen Toleranzprüfung direkt. Wird Toleranzprüfung nicht direkt unterstützt, kann sie durch Kollisionserkennung auf modifizierten Geometrien ersetzt werden: Stellt man sich den Sicherheitsabstand als eine Hülle im Abstand s um die Geometrien der Objekte herum vor, so genügt es, die ursprünglichen Geometrien der Objekte durch Hüllen im Abstand $s/2$ zu ersetzen; eine Kollision zwischen zwei Objekten entspricht dann einem Unterschreiten des Sicherheitsabstandes.[2]

3.1.5 Hindernisse

Die Hindernisse einer Station wirken sich auf die Kollisionserkennung und dadurch auch auf die Wegplanung aus. Sie unterscheiden sich in dieser Hinsicht nur insofern von den Robotern, als sie unbeweglich sind, und können daher entsprechend den Vereinfachungen bei der Kollisionserkennung in Abschnitt 3.1.2 behandelt werden.

Eine weitere Möglichkeit zum vereinfachten Umgang mit Hindernissen sind Umfahrpunkte: Man gibt für jeden Roboter und jede zu berücksichtigende Strecke (z. B. für alle möglichen Verbindungen zwischen den Schweißpunkten sowie zwischen den Schweißpunkten und der Startkonfiguration des Roboters) eine Liste von Umfahrpunkten[3] vor; diese Listen können leer sein, falls keine Hindernisse

[2] Tests auf Kollision eines Roboters mit sich selbst berücksichtigen im Normalfall benachbarte (nur durch ein Gelenk getrennte) Armglieder nicht.

[3] Genaugenommen handelt es sich nicht um Punkte, sondern um Konfigurationen der Roboter.

zu umfahren sind. Will der Roboter eine solche Strecke zurücklegen, muss er zwischen Beginn und Ende der Strecke die gelisteten Umfahrpunkte abfahren.

Da die Umfahrpunkte im Voraus berechnet werden, können sie nur statische Hindernisse berücksichtigen, nicht aber die Bewegungen der anderen Roboter. Die Erstellung der Umfahrpunkte kann daher durch Wegplanungsalgorithmen für einen Roboter und statische Umgebungen (siehe Abschnitt 2.2) erfolgen.

3.1.6 Schweißzeiten

Zur Vereinfachung kann man einheitliche Schweißzeiten verwenden oder ganz auf Schweißzeiten verzichten.

3.1.7 Gegenseitige Beeinflussung der Roboter

Verzichtet man auf Kollisionserkennung, entfällt auch das Problem der gegenseitigen Beeinflussung der Roboter, da diese nicht feststellbar ist. Berücksichtigt man Kollisionen, sind mehrere Vereinfachungsmöglichkeiten denkbar. Sie basieren auf der Beobachtung, dass sich die Roboter nur dann gegenseitig beeinflussen, wenn sie einander nahe sind:

Ausschluss der Beeinflussung Lösungen, bei denen sich die Roboter in ihren Wegen gegenseitig beeinflussen, werden ausgeschlossen. Eine mögliche Definition gegenseitiger Beeinflussung ist, dass sich zwei Roboter genau dann gegenseitig beeinflussen, wenn ihre direkten Wege eine Kollision verursachen. Unter direktem Weg verstehen wir dabei die Ausgabe eines lokalen Wegplaners ohne Berücksichtigung anderer Roboter.

Bestrafung der Annäherung Als weiteres Optimierungskriterium wird die Maximierung des gegenseitigen Abstands der Roboter hinzu genommen, z. B. des Minimums der paarweisen Abstände aller Roboter über die Zeit.

Partitionierung Der Raum wird in m zusammenhängende Gebiete partitioniert; jeder Roboter bearbeitet nur die Schweißpunkte in seinem Gebiet. Die Gebiete sind so zu wählen, dass die Roboter sich nicht oder nur wenig gegenseitig beeinflussen können. Gesucht ist diejenige Partitionierung des Raums, welche die beste Lösung enthält.

Fordert man Kollisionsfreiheit, so sind bei allen Vereinfachungen, die eine gegenseitige Beeinflussung zulassen, Strategien zur Koordination mehrerer Roboter notwendig. Diese gehören zum Problem der Wegplanung.

3.1.8 Unterschiedliche Wege und Wegzeiten

Unterschiedliche Wege und Wegzeiten können zwei Ursachen haben: Die in Abschnitt 1.3.2 beschriebenen unterschiedlichen Bewegungen der verschiedenen Roboter und die gegenseitige Beeinflussung der Roboter.[4] Wir berücksichtigen in diesem Abschnitt nur die unterschiedlichen Bewegungen.

Will man pro Strecke nur mit einer Wegzeit für alle Roboter arbeiten, kann man nach folgendem Schema vorgehen: Man sortiert die in Frage kommenden Roboter nach ihrer Wegzeit, wählt in der Liste der sortierten Roboter eine Trennstelle und schließt alle Roboter mit längerer Wegzeit von der Strecke aus. Die gemeinsame Wegzeit für die Strecke ist dann die Wegzeit des langsamsten zugelassenen Roboters. Die anderen Roboter können die Restzeit z. B. an einem der Schweißpunkte warten oder die Strecke langsamer zurücklegen.

[4] Berücksichtigt man Kollisionen, können sich die Roboter gegenseitig in ihren Wegen beeinflussen, d. h. abhängig von den Bewegungen der anderen Roboter können sich die Wege und Wegzeiten eines Roboters zwischen gleichen Schweißpunkten unterscheiden.

Seien $w_1, \ldots, w_k \in \mathbb{R}_{>0}$, $0 < w_1 \leq w_2 \leq \cdots \leq w_k$, $1 \leq k \leq m$ die Wegzeiten zwischen den beiden Schweißpunkten. Dies können weniger als m Werte sein, da möglicherweise nicht alle Roboter die beiden Schweißpunkte erreichen oder bearbeiten können. Der Ausschluss kann anhand verschiedener Kriterien erfolgen, von denen wir einige vorstellen:

Maximum: $\{i \mid w_i \leq w_k\}$. Der langsamste Roboter bestimmt das Tempo. Alle Roboter werden für die Strecke zugelassen, allerdings auf Kosten der Wegzeit.

Minimum: $\{i \mid w_i = w_1\}$. Nur die schnellsten Roboter werden zugelassen, alle anderen ausgeschlossen. Das restriktivste Kriterium.

Prozentuelle Abweichung: $\left\{i \;\middle|\; w_i \leq w_1(1 + \frac{x}{100})\right\}$ für $0 \leq x \leq 100$. Es werden nur Roboter zugelassen, deren Wegzeit um höchstens $x\%$ von der schnellsten Wegzeit abweicht.

Mittelwert: $\left\{i \;\middle|\; w_i \leq \frac{1}{k}\sum_{j=1}^{k} w_j\right\}$. Das arithmetische Mittel als Trennstelle.

Median: $\{i \mid 1 \leq i \leq x\}$ für $x \in \{1, \ldots, k\}$. Nur die x schnellsten Roboter werden zugelassen. Hierbei sollte x in Abhängigkeit von k gewählt werden, z. B. $x = \lceil \frac{k}{2} \rceil$.

Prinzipiell ist bei der Wahl des Ausschlusskriteriums zwischen Geschwindigkeit und Flexibilität, d. h. einer schnellen Wegzeit und der Anzahl der zugelassenen Roboter, abzuwägen. Eine noch stärkere Vereinfachung ist es, die Wegzeiten gemäß einer nur von den Schweißpunkten abhängigen Metrik festzusetzen, etwa dem euklidischen Abstand.

3.1.9 Wegplanung und Ansteuerung

Bei den möglichen Positionierungen der Schweißzange eines Roboters auf einem Schweißpunkt kann man sich zur Vereinfachung auf eine vorgegebene (kleine) Auswahl an Konfigurationen beschränken, d. h. man gibt $\mathcal{V}_{i,j}$ für $1 \leq i \leq m$, $1 \leq j \leq n$ vor. Im einfachsten Fall ist $|\mathcal{V}_{i,j}| = 1$.

In den beiden Formulierungen des Schweißzellenproblems in Kapitel 1 wird kein explizites Modell für die Ansteuerung der Roboter vorgegeben, sondern lediglich die Existenz einer Menge $\mathcal{M}$ von möglichen Bewegungen vorausgesetzt. Für konkrete Anwendungen ist die Festlegung auf ein Ansteuerungsmodell notwendig. Da die Größe von $\mathcal{M}$ mit der Größe des Suchraums korreliert, ist die Wahl von $\mathcal{M}$ wesentlich für die Komplexität des resultierenden Problems.

3.1.10 Modellierung der Roboter

Zu Vereinfachungen im Zusammenhang mit der Kollisionserkennung siehe Abschnitt 2.1. Die dynamischen Elemente eines Roboters wie z. B. Kabelstränge können ignoriert oder statisch approximiert werden, da ihr Anteil an der Robotergeometrie gering ist und berechnete Lösungen vor ihrem Einsatz getestet bzw. überprüft und gegebenenfalls korrigiert werden können.

Eine stärkere Vereinfachung ist die Verwendung eines niedrigdimensionaleren Konfigurationsraums, d. h. die Arbeit mit stark vereinfachten Robotermodellen. Ein extremer Fall ist die Modellierung der Roboter als Punkte im Raum oder in der Ebene.

3.2 Anforderungen an Algorithmen

Im Hinblick auf die Art der vorzunehmenden Vereinfachungen sind auch die Anforderungen an Algorithmen für das Schweißzellenproblem interessant. Klassische Algorithmen sind meist deterministische, sequenzielle Rechenvorschriften für Registermaschinen zusammen mit einer worst-case Analyse. Diese Sichtweise wird dem vorliegenden Problem weder in Hinsicht auf das Maschinenmodell noch auf die Analyse unbedingt gerecht.

3.2.1 Maschinenmodell

Im vorliegenden Szenario muss der Algorithmus das Problem nicht immer lösen; es genügt, wenn er es mit hoher Wahrscheinlichkeit löst oder es immer löst, aber nur mit hoher Wahrscheinlichkeit auch eine gute Lösung findet. Liefert der Algorithmus kein oder ein zu schlechtes Ergebnis, wird er nochmal ausgeführt – der Optimierungsprozess selbst ist nicht zeitkritisch. Statt deterministischer Maschinenmodelle können randomisierte Algorithmen betrachtet werden. Parallele Verfahren bieten sich aufgrund der zunehmenden Verfügbarkeit von Mehrprozessormaschinen und Rechnerverbundsystemen ebenfalls an.

3.2.2 Analyse

Nicht alle möglichen Probleminstanzen treten in der Praxis tatsächlich auf. So stehen die Roboter nicht alle in derselben Ecke der Station, sondern sind im Allgemeinen eher gleichmäßig um das Werkstück herum angeordnet. Dadurch tritt z. B. das Problem der gegenseitigen Beeinflussung (siehe Abschnitt 1.3.1) nicht in seiner ganzen Schärfe auf. Der Algorithmus muss nicht auf allen Eingaben schnell oder mit gutem Ergebnis arbeiten, sondern nur auf den tatsächlich auftretenden, d. h. es liegt nahe, den Algorithmus auf die Struktur realer Probleminstanzen auszurichten.

Eine worst-case Analyse betrachtet nur die schwierigsten möglichen Eingaben; sie sagt auch nichts über die Häufigkeit dieser Eingaben aus – ihr Anteil an allen Eingaben kann verschwindend gering sein. Der Simplex-Algorithmus ist ein Beispiel für ein Verfahren, das trotz exponentieller worst-case Laufzeit in der Praxis oft und zuverlässig eingesetzt wird.

Bei einer average-case Analyse ist die Wahl der zugrunde gelegten Verteilung ausschlaggebend. In unserem Fall ist unklar, ob die uniforme Verteilung angemessen ist, da über den Anteil und die Verteilung der tatsächlich auftretenden Eingaben innerhalb der möglichen Eingaben nicht genug bekannt ist. Eine Möglichkeit ist die Aufstellung einer Verteilung auf der Basis einer hinreichend großen Anzahl von echten Eingaben.

Die Ergebnisse konventioneller Analysen sind asymptotischer Natur, beziehen sich also auf sehr große Eingabelängen; die praktisch vorliegenden Eingaben befinden sich im Bereich von meist ein bis sechs, höchstens zwölf Robotern und meist um die 100, höchstens 1000 Schweißpunkten. Für solche Eingabegrößen können exponentielle Laufzeiten akzeptabel sein. In diesem Bereich spielen konstante Faktoren, die in der O-Notation ignoriert werden, eine wichtige Rolle.

Zusammenfassend sind Vereinfachungen statthaft, die zu Algorithmen mit hoher worst-case oder average-case Komplexität führen, insofern diese auf tatsächlich auftretenden Eingaben akzeptable Laufzeiten aufweisen; dabei sind randomisierte und parallele Algorithmen zulässig.

3.3 Vereinfachte Problemvarianten

Wir entwerfen drei unterschiedlich stark vereinfachte Varianten des Schweißzellenproblems. Zwei davon reduzieren das Schweißzellenproblem auf den Aspekt der kombinatorischen Optimierung; dementsprechend stark fallen die vorgenommenen Veränderungen aus. Die dritte Variante legt ein allgemeines, kinematisch motiviertes Ansteuerungsmodell zugrunde und bleibt, von kleineren Einschränkungen abgesehen, nahe an den vollständigen Problemdefinitionen.

3.3.1 Variante WCP-A

Wir beginnen mit der am stärksten vereinfachten Variante; sie reduziert das Schweißzellenproblem auf den Kern des in ihm enthaltenen kombinatorischen Optimierungsproblems und bildet die Arbeitsgrundlage für die weiteren Kapitel.

3.3.1.1 Definition

Wir verzichten vollständig auf Kollisionsfreiheit, Nebenbedingungen und Schweißzeiten (gleiche Schweißzeiten mit Wert 0). Als Wege lassen wir nur die direkten Strecken zwischen den Schweißpunkten sowie zwischen den Schweißpunkten und den Startpositionen der Roboter zu; die Wegzeiten setzen wir einheitlich auf den euklidischen Abstand.[5]

Die Aspekte Kollisionserkennung, Wegplanung und Programmierung der Roboter entfallen komplett. Wir interpretieren die Schweißpunkte und die Startpositionen der Roboter als Knoten eines Graphen, die einheitlichen Wegzeiten als Kantengewichte und erhalten ein rein kombinatorisches Optimierungsproblem:

Definition 3.1 (WCP-A) Sei $G = (V, E)$ ein ungerichteter, vollständiger Graph mit Kantengewichten $c : E \to \mathbb{R}_{\geq 0}$ sowie $m \in \mathbb{N}$ ausgezeichneten Knoten $s_1, \ldots, s_m \in V$. Finde eine Partitionierung von V in m Mengen $L_1, \ldots, L_m \subseteq V$ sowie eine Anordnung jeder dieser Mengen

$$\sigma_i : \{1, \ldots, |L_i|\} \to L_i, \quad \sigma_i \text{ bijektiv}, \quad 1 \leq i \leq m,$$

so dass gilt

(1) Jede Partition enthält genau einen der ausgezeichneten Knoten; die Anordnung der Partition beginnt mit diesem Knoten:

$$\forall i \in \{1, \ldots, m\} : \sigma_i(1) = s_i.$$

(2) Das Gewicht der längsten durch die Anordnungen implizierten Rundreise

$$\max \left\{ c\big(\{\sigma_i(|L_i|), s_i\}\big) + \sum_{j=1}^{|L_i|-1} c\big(\{\sigma_i(j), \sigma_i(j+1)\}\big) \;\middle|\; 1 \leq i \leq m \right\}$$

ist minimal.

□

Die ausgezeichneten Knoten entsprechen den Startpositionen der Roboter. Man beachte, dass die Forderung, s_i solle der erste Knoten in der i-ten Partition sein, keine Einschränkung darstellt. Gesucht sind m Rundreisen, die zusammen den Graphen überdecken, wobei kein Knoten mehrfach besucht wird. Die Kosten der längsten Rundreise sind zu minimieren.

In den Definitionen 1.5 und 1.6 dürfen die Roboter Schweißpunkte mehrfach besuchen, was wir beim WCP-A durch die Partitionierung des Graphen in Touren ausschließen. Durch den Verzicht auf die Berücksichtigung von Kollisionen stellt dies keine Einschränkung dar, da zu einer optimalen Lösung mit Mehrfachbesuchen stets eine optimale Lösung ohne Mehrfachbesuche existiert.[6] Man beachte, dass dies nur bei Verzicht auf Kollisionserkennung gilt.

Durch die Einschränkung der möglichen Wege auf die Strecken zwischen den Schweißpunkten ist der Weg eines Roboters durch die Angabe der von ihm bearbeiteten Schweißpunkte und einer

[5] An dieser Stelle wären auch andere Distanzmaße möglich; diese könnten z. B. auf den Winkeländerungen in den Gelenken der Roboter oder auf den durch einen Wegplanungsalgorithmus errechneten Wegzeiten basieren. Wir stellen in der Definition des WCP-A selbst keine Anforderungen an die Kantengewichtung; spätere Algorithmen setzen teilweise die Gültigkeit der Dreiecksungleichung voraus.

[6] Man erhält diese durch Entfernen der mehrfach besuchten Knoten: Gegeben sei eine Lösung für das WCP-A, bei der Knoten mehrfach besucht werden dürfen, d. h. anstelle einer Partitionierung der Knoten ist eine Überdeckung der Knoten durch Teilmengen gesucht. Weiterhin seien v_j ein solcher mehrfach besuchter Knoten, M einer der Roboter, die v_j besuchen sowie v_i und v_k die von M vor bzw. nach v_j besuchten Knoten. Entfernt man v_j aus dem Pfad von M, indem man M direkt von v_i nach v_k gehen lässt, so bleibt die Lösung weiterhin legal (da mindestens ein weiterer Roboter v_j bearbeitet) und die Wegzeit von M verkürzt sich, da die Kantengewichte nicht negativ sind. Dadurch bleibt die Länge des längsten Weges, also der Wert der Lösung, entweder gleich oder verkürzt sich ebenfalls.

Bearbeitungsreihenfolge auf diesen vollständig beschrieben. Dies entspricht der Wahl von Bearbeitungsreihenfolgen für Schweißpunkte als Ansteuerungsmodell:

$$\mathcal{M} = \big\{\sigma : \{1, \ldots, |V'|\} \to V' \;\big|\; \sigma \text{ bijektiv} \wedge V' \subseteq V\big\}.$$

Das WCP-A erinnert in seiner Struktur an das TSP mit m Handlungsreisenden. Es unterscheidet sich von ihm in den unterschiedlichen Kantengewichten der ausgezeichneten Knoten und im Optimierungskriterium: Während beim TSP mit m Handlungsreisenden die Gesamtlänge aller Touren zu minimieren ist, wird beim WCP-A die Länge der längsten Tour minimiert.

3.3.1.2 Komplexität

Das TSP ergibt sich als Spezialfall des WCP-A für $m = 1$ und es gilt $TSP \leq_m^p$ WCP-A.[7] Eine nicht deterministische Turingmaschine für das WCP-A rät zuerst eine Lösung und verifiziert diese anschließend in polynomieller Zeit, indem sie die Kosten der geratenen Lösung berechnet und mit der Schranke aus der Eingabe vergleicht; es folgt WCP-A $\in$ NP. Für die Formulierung des WCP-A als Entscheidungsproblem gilt damit

Satz 3.1 Das WCP-A ist als Entscheidungsproblem NP-vollständig. □

Aus dem Satz folgt die Existenz einer Reduktion WCP-A $\leq_m^p$ TSP. Gegen die Nützlichkeit einer solchen Reduktion spricht, dass wir uns vor allem für die optimalen Lösungen selbst interessieren, und weniger dafür, ob der Wert einer solchen Lösung unterhalb einer bestimmten Schranke liegt. Aus einer Viele-Eine-Reduktion $\leq_m^p$ (engl. many-one reduction) ergibt sich aber nicht notwendigerweise eine Reduktion, die optimale Lösungen erhält. Um die Komplexität des WCP-A genauer einschätzen zu können, vergleichen wir die Größe der Suchräume von TSP und WCP-A:

Satz 3.2 (Suchraumgrösse des WCP-A) Sei $n \in \mathbb{N}_0$ die Anzahl der nicht ausgezeichneten Knoten und $m \in \mathbb{N}$ die Anzahl der ausgezeichneten Knoten. Dann hat der Suchraum des WCP-A die Größe

$$\binom{n+m-1}{n} \cdot n! \quad .$$

□

Beweis Zur Bestimmung der Größe des Suchraums zählen wir die Anzahl der Möglichkeiten, n Knoten in m Partitionen aufzuteilen und diese anzuordnen. Man beachte, dass innerhalb einer gegebenen Partition i die zwei Anordnungen $\sigma_i(1), \sigma_i(2), \ldots, \sigma_i(|L_i|)$ und $\sigma_i(|L_i|), \sigma_i(|L_i|-1), \ldots, \sigma_i(1)$ äquivalent sind in dem Sinne, dass sie der gleichen Tour entsprechen und damit auch die gleichen Kosten haben. Wir zählen solche Anordnungen aus zwei Gründen trotzdem einzeln: Zum Einen handelt es sich gemäß Definition 3.1 um formal unterschiedliche Objekte, zum Anderen sind die beiden Anordnungen für schwierigere Varianten des Schweißzellenproblems nicht mehr äquivalent.[8] Wir berücksichtigen dies beim Vergleich mit dem TSP.

Bezeichne $a_{n,m} \in \mathbb{N}$ mit $n \in \mathbb{N}_0$, $m \in \mathbb{N}$ die Anzahl der gesuchten Möglichkeiten. Wir stellen eine Rekursionsgleichung für $a_{n,m}$ auf und lösen diese mit Hilfe einer bivariaten exponentiellen Erzeugendenfunktion.

[7] Die Aussagen beziehen sich auf die Formulierungen von TSP und WCP-A als Entscheidungsprobleme. Zu den Grundbegriffen der Komplexitätstheorie wie P, NP, $\leq_m^P$ etc. verweisen wir auf die Literatur [83, 84].

[8] Dies gilt bereits bei der Berücksichtigung von Nebenbedingungen bezüglich der Reihenfolge (siehe WCP-B auf Seite 43). In allen Varianten des Schweißzellenproblems, in denen Kollisionen berücksichtigt werden, spielt die Richtung der Tour (die Reihenfolge der Schweißpunkte) eine Rolle.

Im Fall $m = 1$ sind alle n Knoten in einer Partition enthalten und $a_{n,1}$ ist die Anzahl der möglichen Anordnungen von n Elementen. Diese entspricht der Anzahl der Permutationen der Länge n.[9]

Für den Fall $m > 1$ betrachten wir die erste Partition. In dieser können sich zwischen 0 und n Knoten befinden; da die Knoten unterscheidbar sind, gibt es für i Knoten $\binom{n}{i}$ Möglichkeiten, diese aus allen n Knoten auszuwählen. Jede Auswahl kann auf $i!$ Weisen angeordnet werden. Die restlichen $n - i$ Knoten verteilen sich auf die restlichen $m - 1$ Partitionen; dafür gibt es $a_{n-i,m-1}$ Möglichkeiten. Wir erhalten

$$a_{n,m} := \begin{cases} n! & \text{falls } m = 1, \\ \sum_{i=0}^{n} \binom{n}{i} i!\, a_{n-i,m-1} & \text{falls } m > 1 \end{cases}$$

und setzen dies in die entsprechende bivariate, in n exponentielle Erzeugendenfunktion ein:

$$\begin{aligned}
A(y,z) &= \sum_{n\geq 0}\sum_{m\geq 1} a_{n,m}\frac{y^m z^n}{n!} \\
&= \sum_{n\geq 0}\left(n!\frac{y^1 z^n}{n!} + \sum_{m\geq 2}\left(\sum_{i=0}^{n}\binom{n}{i} i!\, a_{n-i,m-1}\right)\frac{y^m z^n}{n!}\right) \\
&= \frac{y}{1-z} + \sum_{n\geq 0}\sum_{m\geq 1}\sum_{i=0}^{n}\binom{n}{i} i!\, a_{n-i,m}\frac{y^{m+1} z^n}{n!} \qquad (3.1) \\
&= \frac{y}{1-z} + y\sum_{m\geq 1} y^m \sum_{n\geq 0}\sum_{i=0}^{n}\binom{n}{i} i!\, a_{n-i,m}\frac{z^n}{n!} \\
&= \frac{y}{1-z} + y\sum_{m\geq 1} y^m \left(\sum_{n\geq 0} n!\frac{z^n}{n!}\right)\left(\sum_{n\geq 0} a_{n,m}\frac{z^n}{n!}\right) \qquad (3.2) \\
&= \frac{y}{1-z} + \frac{y}{1-z}\sum_{n\geq 0}\sum_{m\geq 1} a_{n,m}\frac{y^m z^n}{n!} \\
&= \frac{y}{1-z}\big(1 + A(y,z)\big)\ .
\end{aligned}$$

Dabei wurde in 3.1 und 3.2 die Summenidentität $\sum_{n\geq 0} z^n = \frac{1}{1-z}$ und in 3.2 die Binomialkonvolution

$$\left(\sum_{n\geq 0} b_n\frac{z^n}{n!}\right)\cdot\left(\sum_{n\geq 0} c_n\frac{z^n}{n!}\right) = \sum_{n\geq 0}\sum_{k=0}^{n}\binom{n}{k} b_k c_{n-k}\frac{z^n}{n!}$$

verwendet. Durch Umstellen erhalten wir die Funktionalgleichung der Erzeugendenfunktion:

$$\begin{aligned}
A(y,z) &= \frac{y}{1-z}\big(1 + A(y,z)\big) \\
\iff \quad A(y,z)\Big(1 - \frac{y}{1-z}\Big) &= \frac{y}{1-z} \\
\iff \quad A(y,z) &= \frac{y}{1-z-y} \quad .
\end{aligned}$$

[9] Für die erste Stelle der Permutation stehen n Elemente zur Verfügung, für die zweite Stelle $n-1$ usw. Welche Elemente in einem Schritt zur Verfügung stehen, hängt von allen vorher gewählten Elementen ab. Es ergeben sich $n \cdot (n-1) \cdots 2 \cdot 1 = n!$ Möglichkeiten.

Entwicklung um $y = z = 0$ liefert

$$\begin{aligned} A(y,z) &= y\,\frac{1}{1-(z+y)} \\ &= y\sum_{k\geq 0}(z+y)^k \qquad (3.3) \\ &= \sum_{k\geq 0}\sum_{i=0}^{k}\binom{k}{i}z^i y^{k-i+1} \qquad (3.4) \\ &= \sum_{k\geq 0}\sum_{i=0}^{k}\frac{k!}{(k-i)!}\frac{y^{k-i+1}z^i}{i!} \end{aligned}$$

unter der Verwendung der geometrischen Reihe in 3.3 und des binomischen Lehrsatzes in 3.4. Mit $i = n$ und $k = m + n - 1$ ergeben sich die gesuchten Koeffizienten zu

$$\left[\frac{y^m z^n}{n!}\right] A(y,z) = \frac{(m+n-1)!}{(m-1)!} = \binom{n+m-1}{n}\cdot n! \quad .$$

□

Für einen Graphen mit $k \geq 3$ Knoten hat das TSP einen Suchraum der Größe $(k-1)!/2$, da der erste Knoten der Tour beliebig gewählt werden kann (das TSP kennt keinen Startknoten) und Touren, die sich nur in ihrer Orientierung unterscheiden, als äquivalent angesehen werden können. Wir zählen solche Touren dennoch einzeln, um einen angemessenen Vergleich mit der Suchraumgröße des WCP-A zu ermöglichen. Es ergibt sich ein Suchraum der Größe $(k-1)!$ für das TSP.

Fasst man die Anzahl $|V|$ der Knoten des Graphen, also die Summe der Anzahl der gewöhnlichen und der Anzahl der ausgezeichneten Knoten, $n + m$, als Eingabegröße für das WCP-A auf, ergibt sich als Quotient der Suchraumgrößen

$$\frac{\text{Suchraumgröße WCP-A}}{\text{Suchraumgröße TSP}} = \frac{(n+m-1)!n!}{n!(m-1)!(n+m-1)!} = \frac{1}{(m-1)!}. \qquad (3.5)$$

Fasst man nur die Anzahl der gewöhnlichen Knoten als Eingabegröße für das WCP-A auf, ergibt sich folgendes Verhältnis:

$$\frac{\text{Suchraumgröße WCP-A}}{\text{Suchraumgröße TSP}} = \frac{(n+m-1)!n!}{n!(m-1)!(n-1)!} = \binom{n+m-1}{n}\cdot n. \qquad (3.6)$$

Je nachdem, wie man den Begriff der Eingabegröße für das WCP-A definiert, wächst dessen Suchraum stärker oder schwächer als der Suchraum des TSP. Die Ursache hierfür liegt im Verhalten der Fakultät: Schreibt man die Größe des Suchraums des WCP-A als $(n+m-1)!/(m-1)! = (n+m-1)\cdots m$, sieht man, dass sich der Suchraum des TSP entweder nur in den kleinsten $m-1$ Faktoren oder zusätzlich auch in den m größten Faktoren unterscheidet. Letztere dominieren den Wert des Ausdrucks.

In unserem Fall ist $m \ll n$. Tatsächlich gilt sogar $m \leq 12$ (siehe Abschnitt 3.2.2), d. h. wir können m als konstant annehmen. Unter dieser Annahme ist der Suchraum des WCP-A in 3.5 um einen konstanten Faktor kleiner als der Suchraum des TSP; in 3.6 ist er um nicht mehr als einen polynomiellen Faktor größer:

$$\begin{aligned}\frac{(n+m-1)!}{n!(m-1)!}n &= \frac{1}{(m-1)!}(n+m-1)\cdots n \leq \frac{1}{(m-1)!}(n+m-1)^m \\ &= \frac{1}{(m-1)!}\sum_{i=0}^{m}\binom{m}{i}n^i(m-1)^{m-i} \leq \frac{(m-1)^m}{(m-1)!}\sum_{i=0}^{m}\frac{m!}{i!(m-i)!}n^i \\ &\leq \frac{(m-1)^m m!}{(m-1)!}\sum_{i=0}^{m}n^i = \underbrace{(m-1)^m m}_{\text{konstant}}\sum_{i=0}^{m}n^i = O\left(n^m\right).\end{aligned}$$

Die Suchraumgröße des WCP-A unterscheidet sich damit von der Suchraumgröße des TSP um höchstens einen polynomiellen Faktor.

Die naive Vorgehensweise zur Lösung des WCP-A mit Hilfe des TSP, die Enumeration aller m^n möglichen Partitionierungen und das Lösen der entsprechenden TSP-Instanzen für jede entstehende Partition, erfordert exponentiell viele Aufrufe des TSP. Wir wenden uns daher in Kapitel 4 einem anderen Ansatz zu.

3.3.2 Variante WCP-B

Wir vereinfachen wie im vorherigen Abschnitt mit dem Ziel eines rein kombinatorischen Optimierungsproblems, übernehmen diesmal aber mehr Anforderungen aus den vollständigen Formulierungen des Schweißzellenproblems. Dazu verzichten wir weiterhin auf Kollisionsfreiheit und andere Wege als die direkten Strecken zwischen den Schweißpunkten sowie zwischen den Schweißpunkten und den Startpunkten der Roboter, d. h. wir bleiben bei Bearbeitungsreihenfolgen als Ansteuerungsmodell. Dagegen erlauben wir Nebenbedingungen, individuelle Schweißzeiten sowie unterschiedliche Wegzeiten der Roboter.

Durch die Zulassung unterschiedlicher Wegzeiten können die tatsächlichen (nicht durch Kollisionen eingeschränkten) Wegzeiten der Roboter[10] verwendet werden. Als zentrale Einschränkung verbleibt der Verzicht auf die Kollisionsfreiheit und die damit verbundene Wegplanung.

Wie beim WCP-A interpretieren wir die Schweißpunkte und die Startkonfigurationen der Roboter als Knoten eines ungerichteten Graphen, verzichten jedoch auf dessen Vollständigkeit. Dadurch können Roboter von einzelnen Strecken ausgeschlossen werden, etwa weil sie diese nicht zurücklegen können. Durch entsprechende Wahl der Kanten und Wegkosten lassen sich so Hindernisse indirekt berücksichtigen. Man beachte, dass die Forderung nach Vollständigkeit des Graphen nicht wesentlich ist.[11]

Jeder Kante werden m Gewichte zugeordnet, wobei das i-te Gewicht zur Berechnung der Kosten der i-ten Tour verwendet wird. Die Schweißzeiten ordnen wir den Knoten zu; sie gehen ebenfalls in die Berechnung der Tourkosten ein, sind aber unabhängig von den Robotern.

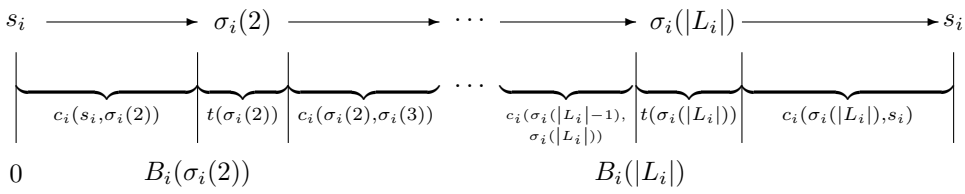


Abbildung 3.1: Zusammensetzung des Weges von M_i in Definition 3.2

[10] Diese Wegzeiten können durch die Steuersoftware der Roboter ermittelt werden. Stehen Umfahrpunkte zur Verfügung, können auf diesem Weg auch statische Hindernisse berücksichtigt werden.

[11] Siehe die Verwendung großer Kantengewichte in Abschnitt 2.3.4.1.

DEFINITION 3.2 (WCP-B) Sei $G = (V, E)$ ein ungerichteter Graph mit den Knoten $V = V' \cup V''$; dabei sei $V' = \{1, \ldots, n\}$ und $V'' = \{s_1, \ldots, s_m\}$. Den Knoten in V' seien durch $t : V' \to \mathbb{R}_{\geq 0}$ Schweißzeiten zugeordnet. Weiterhin seien m Kantengewichtungen $c_i : E \to \mathbb{R}_{\geq 0}$, $1 \leq i \leq m$ sowie Nebenbedingungen $C_E \subseteq \{1, \ldots, m\} \times V'$ bezüglich des Ausschlusses und Nebenbedingungen $C_S \subseteq V' \times V'$ bezüglich der Reihenfolge gegeben.

Finde eine Partitionierung von V in m Mengen $L_1, \ldots, L_m \subseteq V$ sowie eine Anordnung jeder dieser Mengen

$$\sigma_i : \{1, \ldots, |L_i|\} \to L_i, \quad \sigma_i \text{ bijektiv}, \quad 1 \leq i \leq m,$$

so dass gilt

(1) Jede Partition enthält genau einen der ausgezeichneten Knoten; die Anordnung der Partition beginnt mit diesem Knoten:

$$\forall i \in \{1, \ldots, m\} : \sigma_i(1) = s_i.$$

(2) Das Gewicht der längsten durch die Anordnungen implizierten Rundreise

$$\max \left\{ c_i\big(\{\sigma_i(|L_i|), s_i\}\big) + \sum_{j=1}^{|L_i|-1} c_i\big(\{\sigma_i(j), \sigma_i(j+1)\}\big) + t\big(\sigma_i(j+1)\big) \;\middle|\; 1 \leq i \leq m \right\}$$

ist minimal.

(3) Die Nebenbedingungen sind erfüllt. Es bezeichne $B : V' \to \mathbb{R}_{\geq 0}$ den Zeitpunkt des Beginns der Bearbeitung eines Knotens.[12]

   (a) Reihenfolge: $\forall (i, j) \in C_S : B(i) + t(i) < B(j)$
   (b) Ausschluss: $\forall (i, j) \in C_E : j \notin L_i$

□

Um die Notation im Fall einer bis auf die Startkonfiguration leeren Partition nicht unnötig zu erschweren, gehen wir bei der Berechnung der Wegkosten von $c_i(s_i, s_i) = 0$ aus. Die wesentlichen Änderungen gegenüber dem WCP-A sind die geänderte Berechnung der Kosten für die Touren (Abbildung 3.1 zeigt eine grafische Darstellung der Zusammensetzung des Weges von M_i) und die Einschränkung des Suchraums durch die Nebenbedingungen. Durch letztere sind im Gegensatz zum WCP-A Probleminstanzen möglich, die keine Lösung besitzen.

Das WCP-A ergibt sich als Spezialfall des WCP-B mit identischen Kantengewichtungen, auf Null gesetzten Schweißzeiten und leeren Nebenbedingungen; insbesondere gilt WCP-A $\leq_m^P$ WCP-B. Da eine nicht deterministische Turingmaschine eine Lösung für das WCP-B in polynomieller Zeit raten und verifizieren kann, folgt

SATZ 3.3 Das WCP-B ist als Entscheidungsproblem NP-vollständig. □

[12] Analog zu Definition 1.6. Formal: Sei $L : V \to \{1, \ldots, m\}$ mit $L(j) = i$ falls $j \in L_i$ die Funktion, die einem Knoten seine Partition zuordnet und σ^{-1} die Umkehrfunktion zu σ. Dann ist

$$B(j) := c_{L(j)}\ \{s_{L(j)}, \sigma_{L(j)}(2)\}\ + \sum_{k=2}^{\sigma_{L(j)}^{-1}(j)-1} c_{L(j)}\ \{\sigma_{L(j)}(k), \sigma_{L(j)}(k+1)\}\ + t\ \sigma_{L(j)}(k)\ .$$

3.3.3 Variante WCP-C

Abschließend entwerfen wir eine Variante des Schweißzellenproblems, die nahe an der Komplexität der vollständigen Problemdefinitionen bleibt. Sie beinhaltet Kollisionserkennung und Wegplanung sowie die Ansteuerung der Roboter und setzt auf einem allgemeinen, kinematisch motivierten Ansteuerungsmodell auf.

Wir gehen von der kontinuierlichen Formulierung des Schweißzellenproblems in Definition 1.5 aus. Die gesuchten Bewegungen der Roboter approximieren wir durch diskrete Folgen von Roboterkonfigurationen; dabei verwenden wir folgendes Ansteuerungsmodell: Die Roboter werden gemäß einem systemweiten, für alle Roboter gleichen Takt angesteuert.[13] Die Konfiguration eines Roboters kann für jeden der durch die Taktung festgelegten Zeitpunkte angegeben werden. Die dadurch implizierten Winkeländerungen dürfen für jedes Gelenk festgesetzte maximale Geschwindigkeiten und Beschleunigungen nicht überschreiten. Bewegungen, d. h. Konfigurationsfolgen, bei denen ein Roboter mit sich selbst kollidiert, sind ebenfalls unzulässig. In diesem Abschnitt bezeichne $\mathcal{M} \subset \mathcal{C}^\infty$ die Menge aller Folgen von Konfigurationen, die diesen Bedingungen genügen.

Die Roboter sind an den Taktzeitpunkten auf Kollisionen mit sich selbst, mit anderen Robotern, mit Hindernissen und mit dem Werkstück zu überprüfen. Die Robotermodelle werden nicht vereinfacht. Die Schweißzeitpunkte dürfen ausschließlich auf den Taktzeitpunkten liegen, die Schweißzeiten müssen Vielfache der Taktzeit sein. Für jeden Roboter M_i und jeden Schweißpunkt J_j geben wir eine endliche Menge $\mathcal{V}_{i,j}$ an möglichen Schweißstellungen vor.

Alle aufgestellten Forderungen beziehen sich nicht auf absolute Zeitpunkte, sondern auf Übereinstimmungen mit den Taktzeitpunkten. Dadurch ist die Taktzeit selbst unerheblich und wir können uns bei der Angabe von Zeiten auf den Index des Taktzeitpunkts, d. h. die natürlichen Zahlen, beschränken. Die Schweißzeiten seien in der folgenden Definition als $t_1, \ldots, t_n \in \mathbb{N}_0$ gegeben. Ist $s_j \in \mathbb{N}_0$ der Schweißzeitpunkt von J_j, so ist der bearbeitende Roboter ab dem Zeitpunkt $s_j + t_j$, ausschließlich des Zeitpunkts selbst, wieder verfügbar.

Definition 3.3 (WCP-C) Gegeben Station, Schweißpunkte und Nebenbedingungen gemäß Definition 1.5 und Abschnitt 3.3.3. Finde m Folgen von Konfigurationen $f_1, \ldots, f_m \in \mathcal{M}$ und $t_{\max} \in \mathbb{N}_0$ sowie n Schweißzeitpunkte $s_j \in \{0, \ldots, t_{\max} - t_j\} \subset \mathbb{N}_0, 1 \leq j \leq n$, so dass die folgenden Bedingungen erfüllt sind:

(1) *Optimalität:* Die Gesamtbearbeitungszeit muss minimal sein:

$$t_{\max} = \min \left\{ t \in \mathbb{N}_0 \;\middle|\; \begin{array}{l} \exists g_1, \ldots, g_m \in \mathcal{M}\, \exists r_1 \in [0, t - t_1], \ldots, r_n \in [0, t - t_n] : \\ g_1, \ldots, g_m, r_1, \ldots, r_n, t \text{ erfüllen die Bedingungen 2,3,4,5} \end{array} \right\}$$

(2) *Vollständigkeit:* Am Ende müssen alle Schweißpunkte bearbeitet worden sein:

$$\forall j \in \{1, \ldots, n\}\, \exists i \in a_j : f_i(\{s_j, \ldots, s_j + t_j\}) = v \in \mathcal{V}_{i,j}$$

(3) *Kollisionsfreiheit:* Kein Unterschreiten des Sicherheitsabstandes zwischen Robotern, Bauteil und Station:

$$\forall t \in [0, t_{\max}] \subset \mathbb{N}_0\, \forall i, j \in \{1, \ldots, m\}, i \neq j : \Big(d_{i,j}\big(f_i(t), f_j(t)\big) > s \Big) \wedge \Big(d_i\big(f_i(t)\big) > s \Big)$$

(4) *Nebenbedingungen:* Die Nebenbedingungen bezüglich der Reihenfolge sind erfüllt:

$$\forall (i, j) \in C_S : s_i + t_i < s_j$$

[13] Ein Beispiel für eine mögliche Taktzeit sind 12 ms.

(5) Jeder Roboter beginnt und beendet seine Bewegung in seiner Ausgangsstellung:

$$\forall i \in \{1, \ldots, m\} : f_i(0) = f_i(t_{\max}) = S_i \quad \wedge \quad f_i'(0) = f_i'(t_{\max}) = \vec{0}$$

□

Durch die Verwendung von $\mathcal{M}$ wird sichergestellt, dass die Roboter nicht mit sich selbst kollidieren können. Die Einhaltung der Ausschluss-Nebenbedingungen erfolgt implizit durch die Verwendung der a_j in Punkt 2.

Da nur zu den Taktzeitpunkten auf Kollisionen geprüft wird, besteht die Möglichkeit, dass Kollisionen der Roboter zwischen den Taktzeitpunkten unerkannt bleiben. Da die Geschwindigkeit der Roboter beschränkt ist, lässt sich dies durch die Wahl einer hinreichend kleinen Taktzeit verhindern. Ist die Taktzeit vorgegeben, können zusätzliche Kollisionstests an interpolierten Konfigurationen zwischen den Taktzeitpunkten durchgeführt werden.

Bezüglich der Berechnung der Geschwindigkeit f' und der Beschleunigung f'' in den Gelenken eines Roboters ist zu beachten, dass es sich bei f nicht um eine kontinuierliche Funktion, sondern um eine diskrete Folge von Konfigurationen handelt.[14]

Das WCP-C ist im Wesentlichen eine auf einer Diskretisierung der Zeit basierende Variante der kontinuierlichen Formulierung.

3.4 Weiterführendes

Wir haben in diesem Kapitel drei vereinfachte Varianten des Schweißzellenproblems eingeführt: Die am stärksten vereinfachende Variante ist das WCP-A. Sie reduziert das Schweißzellenproblem auf ein rein kombinatorisches Optimierungsproblem: Die Verteilung der Schweißpunkte auf die Roboter und die Bestimmung von Bearbeitungsreihenfolgen für diese. Obwohl es strukturelle Ähnlichkeiten zum TSP mit m Handlungsreisenden aufweist, unterscheidet es sich von diesem im Optimierungskriterium und durch die unterschiedlichen Startknoten. Das WCP-A bildet die Arbeitsgrundlage für die weiteren Kapitel.

Die zweite Variante, das WCP-B, nähert sich dem ursprünglichen Schweißzellenproblem so weit wie möglich, ohne Kollisionen und die damit verbundene Wegplanung hinzunehmen zu müssen. Die wesentlichen Unterschiede zum WCP-A sind die Verwendung eigener Kantengewichte für jede Partition und die Berücksichtigung der Nebenbedingungen. Die unterschiedlichen Wegzeiten der Roboter ändern die Berechnung der Tourlängen, nicht jedoch die Größe des Suchraums. Die Nebenbedingungen verringern die Anzahl zulässiger Lösungskandidaten, schränken den Suchraum jedoch nicht generell ein.[15] Wir gehen in dieser Arbeit nicht weiter auf das WCP-B ein, jedoch lassen sich die meisten in den Kapiteln 4 und 5 vorgestellten Algorithmen für das WCP-A mit geringfügigen Änderungen auch für das WCP-B verwenden.

Sowohl WCP-A als auch WCP-B beruhen auf Bearbeitungsreihenfolgen von Schweißpunkten als Ansteuerungsmodell; beide Varianten vereinfachen das Schweißzellenproblem sehr stark. Im Gegensatz dazu handelt es sich bei der dritten Variante WCP-C um eine — bis auf geringfügige Einschränkungen bei den Schweißstellungen — direkte Umsetzung von Definition 1.5 auf Basis einer Diskretisierung der Zeit. Als Ansteuerungsmodell werden kinematisch restringierte Konfigurationsfolgen verwendet. Mit dem WCP-C soll ein realistisches, aber dennoch handhabbares Modell für

[14] Steht der Roboter an den Taktzeitpunkten still, genügt die Kenntnis von Taktzeit sowie maximaler Geschwindigkeit und Beschleunigung, um festzustellen, ob der Roboter eine Strecke zwischen zwei Taktzeitpunkten zurücklegen kann. Ist der Roboter an den Taktzeitpunkten in Bewegung, müssen Geschwindigkeit und Beschleunigung zu Beginn einer solchen Strecke berücksichtigt werden.

[15] Die Nebenbedingungen sind Teil der Eingabe und schränken daher immer nur den Suchraum einer bestimmten Probleminstanz ein. Insbesondere gibt es Probleminstanzen mit $C_S = C_E = \emptyset$.

das Schweißzellenproblem vorgestellt werden. Es enthält keine praktisch relevanten Einschränkungen und verwendet ein realistisches Ansteuerungsmodell. Das WCP-C kann als Ausgangspunkt für weitere Untersuchungen zum Schweißzellenproblem dienen.

Auf der Basis anderer Ansteuerungsmodelle oder Vereinfachungen sind weitere Varianten des Schweißzellenproblems möglich. Denkbar wäre z. B. eine kontinuierliche Variante mit einem Ansteuerungsmodell auf der Basis (stückweise) stetiger Funktionen, z. B. Konkatenationen von durch Stützstellen oder Koeffizienten beschriebenen Polynomen. Eine weitere Möglichkeit wäre ein auf Steuerbefehlen basierendes Ansteuerungsmodell, bei dem die Bewegungen der Roboter durch Folgen von Befehlen zu (relativen) Konfigurationsänderungen repräsentiert werden.

Kapitel 4

Ein evolutionärer Ansatz

Wir stellen in diesem Kapitel ein flexibles evolutionäres Rahmenwerk für Algorithmen zur Lösung des WCP-A vor. Nach einer Begründung der Entscheidung für einen evolutionären Ansatz rekapitulieren wir kurz einige wichtige Begriffe der evolutionären Optimierung und diskutieren einige ihrer spezielleren Aspekte. Anschließend stellen wir den evolutionären Ansatz für das WCP-A im Detail vor. Die weitgehende Parametrisierbarkeit des Ansatzes ermöglicht nicht nur die Abbildung der eigentlichen evolutionären Algorithmen, sondern auch diverser anderer Verfahren aus der stochastischen Optimierung, u. a. simuliertes Abkühlen und iterierte lokale Suche.

Da evolutionäre Algorithmen in der Literatur ausgiebig behandelt werden [48, 47], setzen wir voraus, dass der Leser mit den grundlegenden Ideen bereits vertraut ist und beschränken uns auf eine kurze Darstellung der für uns relevanten Aspekte.

4.1 Wahl des Optimierungsverfahrens

Wir begründen im Folgenden unsere Entscheidung für einen evolutionären Ansatz zur Lösung des WCP-A. Dabei gehen wir von den in Abschnitt 2.3.3 vorgestellten Methoden zur globalen Optimierung aus. Tabelle 4.1 gibt einen Überblick über alle Verfahren und ihre Eignung für das Schweißzellenproblem.

Da es sich um ein kombinatorisches Optimierungsproblem handelt, entfallen alle Methoden, bei denen die Zielfunktion als geschlossener algebraischer Ausdruck vorliegen muss (Intervall-Methoden, Fortsetzungs- und Glättungsmethoden). Statistische Methoden sind besser für kontinuierliche Optimierungsprobleme geeignet; zudem ist die Auswertung der Zielfunktion nicht teuer genug, um ihren Einsatz sinnvoll erscheinen zu lassen.

Die Größe des Suchraums des WCP-A wächst extrem schnell an (siehe Satz 3.2). Obwohl mittlerweile größere Instanzen von vergleichbaren Problemen wie dem TSP exakt gelöst werden können (siehe Fußnote auf Seite 32), erfordert dies beachtliche Rechenzeiten und einen erheblichen Aufwand bei Entwurf und Implementierung der entsprechenden Algorithmen. Wir betrachten daher optimale enumerative Verfahren (Dynamisches Programmieren, Branch & Bound, Lineare Programmierung, Nichtlineare Programmierung) an dieser Stelle nicht weiter.[1]

Sämtliche verbleibenden Verfahren sind stochastischer und heuristischer Natur. Das Baum-Abkühlen ist ausschließlich für kontinuierliche Optimierungsprobleme entworfen worden. Multistart- und Clustering-Methoden sind nicht für hochdimensionale Probleme mit vielen lokalen Op-

[1] Kleinere Probleminstanzen des WCP-A lassen sich bei vertretbarem Implementierungsaufwand mit diesen Methoden lösen. Um größere Instanzen mit linearer Programmierung lösen zu können, sind wie beim TSP genauere Kenntnisse über die Struktur des durch die entsprechenden Ungleichungen für das WCP-A beschriebenen Polyeders nötig.

	Optimierungsmethode	Kommentar
×	Dynamisches Programmieren	Suchraum zu groß
×	Branch & Bound, (Nicht)lineare Programmierung	Suchraum bzw. Aufwand für Entwurf und Implementierung zu groß
×	Intervall-Methoden, Fortsetzungs- und Glättungsmethoden	Zielfunktion muss als geschlossener Ausdruck vorliegen
×	Multistart und Clustering	Nicht für hochdimensionale Probleme mit vielen lokalen Optima geeignet
×	Statistische Methoden	Besser für kontinuierliche Probleme geeignet. Auswertung der Zielfunktion nicht teuer genug
×	Baum-Abkühlen	Nur für kontinuierliche Probleme geeignet
✓	Ameisen-Algorithmen, Tabu-Suche	Geeignet
✓	Metropolis-Algorithmus	Spezialfall des simulierten Abkühlens
✓	Simuliertes Abkühlen, Memetische Algorithmen	Spezialfall evolutionärer Algorithmen
✓	Scatter Search, Kontrollierte Zufällige Suche	Variante evolutionärer Algorithmen
✓	Grasp, Iterierte Lokale Suche	Abbildbar durch evolutionäre Algorithmen
✓	Evolutionäre Algorithmen	Geeignet

Tabelle 4.1: Eignung von Optimierungsverfahren für das WCP-A

tima und Nebenbedingungen[2] geeignet. Der Metropolis-Algorithmus ist ein Spezialfall des simulierten Abkühlens. Das simulierte Abkühlen und die memetischen Algorithmen sind Spezialfälle der evolutionären Algorithmen; Scatter Search und die kontrollierte zufällige Suche sind Varianten der evolutionären Algorithmen.

Die übrigen Verfahren — evolutionäre Algorithmen, Ameisen-Algorithmen, Tabu-Suche, Grasp und iterierte lokale Suche — eignen sich zur Lösung des WCP-A. Wir haben uns aus den folgenden Gründen für einen evolutionären Ansatz entschieden:

Flexibilität Evolutionäre Algorithmen lassen sich durch problemspezifische Repräsentationen und Operatoren gut an ein Problem anpassen. Sie bieten zahlreiche Möglichkeiten zur Kontrolle und Ausgestaltung des Verfahrens. Andere Methoden können in den evolutionären Ansatz integriert oder durch diesen abgebildet werden.

Bestehende Lösungen Das von der Firma *Beratende Ingenieure Frankfurt* [88] für eine Variante des Schweißzellenproblems entwickelte Programm OptiSpot basiert auf dem Verfahren des simulierten Abkühlens. Letzteres lässt sich als ein Spezialfall evolutionärer Algorithmen auffassen. Dies zeigt die prinzipielle Eignung solcher Verfahren für die vorliegende Problemklasse, bietet die Möglichkeit zum Vergleich und erlaubt die Untersuchung differenzierterer Fragestellungen.

Implementierbarkeit Das Verfahren an sich ist einfach zu implementieren. Daneben gibt es zahlreiche bestehende Programme, Programmbibliotheken und Entwicklungsumgebungen für evolutionäre Algorithmen.

Insbesondere die Abbildbarkeit der meisten anderen Verfahren durch einen hinreichend parametrisierbaren evolutionären Ansatz war für die Entscheidung ausschlaggebend.

[2] Zwar sind Nebenbedingungen kein Bestandteil des WCP-A, wir bevorzugen jedoch Optimierungsmethoden, die sich leicht auf das WCP-B übertragen lassen.

4.2 Grundbegriffe der evolutionären Optimierung

Allen evolutionären Ansätzen gemeinsam ist die Orientierung am natürlichen biologischen Vorgang der Evolution. Dabei wird von diesem abstrahiert, um zu einem Optimierungsverfahren zu gelangen. Die Grundidee ist folgende:

Eine Multimenge von Lösungen wird erstellt. Aus diesen werden neue Lösungen konstruiert, indem bestehende Lösungen abgeändert bzw. zu neuen Lösungen kombiniert werden. Dabei verdrängen bessere Lösungen schlechtere, so dass die Gesamtzahl der in der Multimenge enthaltenen Lösungen über die Zeit konstant bleibt. Dieser Zyklus setzt sich so lange fort, bis eine Abbruchbedingung erfüllt ist.

Abbildung 4.1 zeigt eine grafische Darstellung des Ablaufs. In den folgenden Abschnitten werden die wichtigsten Begriffe der evolutionären Optimierung kurz erläutert.

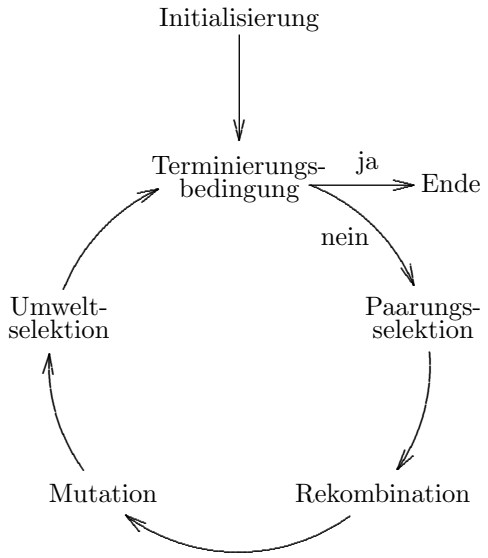


Abbildung 4.1: Ablaufschema evolutionärer Algorithmen. Die Abbildung orientiert sich an der Darstellung im Buch von Weicker [48].

4.2.1 Genotyp, Phänotyp, Individuum

Die Gesamtheit der physischen Merkmale eines Organismus wird in der Biologie als dessen *Phänotyp* bezeichnet, sein gesamtes genetisches Material als *Genom* oder *Genotyp*. Der Phänotyp ist, u. a. aufgrund des Einflusses der Umwelt auf die Genexpression, durch den Genotyp nicht vollständig bestimmt. Einen einzelnen Organismus (Genotyp und Phänotyp) bezeichnen wir als *Individuum*.

In der evolutionären Optimierung entspricht der Genotyp der vom Algorithmus verwendeten Repräsentation eines Individuums, während der Phänotyp einer Lösung des zu optimierenden Problems, also einem Punkt im Suchraum, entspricht. Da die Bewertung (siehe Fitness in Abschnitt 4.2.3) eines Individuums auf Grundlage des Phänotyps erfolgt, wird eine Dekodierungsfunktion benötigt, die jedem Genotyp einen Phänotyp zuordnet. Man beachte, dass mehrere Genotypen auf den gleichen Phänotyp abgebildet werden können. Die Dekodierungsfunktion sollte surjektiv sein. Sind Genotyp und Phänotyp identisch, ist die Dekodierungsfunktion die Identität und eine Unterscheidung zwischen Genotyp und Phänotyp unnötig.

4.2.2 Population, Generation

Der Begriff *Population* bezeichnet in der Biologie eine Gruppe von Organismen derselben Art, die ein bestimmtes (geographisches) Areal bevölkern. In der evolutionären Optimierung bezeichnet der Begriff eine Multimenge von Individuen.

Der Begriff *Generation* steht im Sprachgebrauch für eine Gruppe von Individuen, die in etwa zur selben Zeit geboren wurden. Im Gegensatz dazu versteht man in der evolutionären Optimierung unter Generation die Population zu Beginn jedes Zyklus des Algorithmus (siehe Abbildung 4.1). Das Alter der Individuen spielt dabei keine Rolle.

4.2.3 Selektion, Fitness

Selektion (lat. selectio, Auswahl, Auslese) ist der Vorgang der Auswahl eines Teils einer Population. Selektion tritt im Zyklus an zwei Stellen auf: Umweltselektion und Paarungsselektion. Die Umweltselektion bestimmt, welche Individuen in die nächste Generation übernommen werden; sie dient zur Kontrolle von Größe und Zusammensetzung einer Generation. Die Paarungsselektion wählt die Individuen aus, die zur Rekombination herangezogen werden; sie dient zur Kontrolle über die Zusammensetzung der Kinder.

Die *Fitnessfunktion* ordnet jedem Individuum gemäß seinem Phänotyp einen Wert zu; diesen nennt man die *Fitness* des Individuums. Auf den Fitnesswerten ist meist eine totale, zumindest jedoch eine partielle Ordnung definiert.[3] Hängt die Fitnessfunktion ausschließlich vom Phänotyp des Individuums ab, spricht man von einer *statischen* Fitnessfunktion; *dynamische* Fitnessfunktionen können weitere Größen berücksichtigen, z. B. die Anzahl bereits simulierter Generationen. Die meisten Selektionsmethoden orientieren sich an der Fitness der Individuen.

In der Natur beeinflusst der Zufall das Überleben und die Fortpflanzung eines Individuums unmittelbar und maßgeblich, z. B. durch die Verfügbarkeit von Paarungspartnern, das Auftreten von Nebenbuhlern oder Gefahren, die Tagesform, das Nahrungsangebot usw. Dies spiegelt sich im Algorithmus in den Selektionsmethoden wider, die häufig Zufallsbits verwenden, nicht jedoch in der (deterministischen) Fitnessfunktion. In diesem Sinne entspricht die Fitness eines Individuums seinem Potenzial bezüglich der Fähigkeit zu überleben und sich fortzupflanzen.

4.2.4 Rekombination, Mutation

Reproduktion ist die Erzeugung neuer Individuen aus bestehenden Individuen. Man unterscheidet in der Biologie zwischen sexueller Reproduktion (mehrere Eltern erzeugen mindestens ein Kind) und asexueller Reproduktion (ein Elter erzeugt ein oder mehrere, zu ihm genetisch identische, Kinder). Dies entspricht k-ären, $k \geq 2$, bzw. unären Rekombinationsoperatoren.

Während in der Natur stets nur das genetische Material der Eltern neu kombiniert wird, sind Rekombinationsoperatoren in dieser Hinsicht nicht eingeschränkt; insbesondere kann der Genotyp der Kinder (oder Teile davon) neu konstruiert werden.

Eine *Mutation* ist eine Veränderung im Genotyp eines Individuums. Hat die Mutation keine Auswirkung auf den Phänotyp, spricht man von einer *neutralen* Mutation.

4.3 Anmerkungen

Es folgen einige weitergehende Ausführungen zu Begriffen der evolutionären Optimierung sowie zu Aspekten des Entwurfs evolutionärer Algorithmen.

[3] Die Notwendigkeit einer totalen Ordnung hängt von den Anforderungen der einzelnen Bestandteile des evolutionären Algorithmus ab, u. a. auch von den Selektionsmethoden. Für eine Sortierung der Individuen nach ihrer Fitness ist eine totale Ordnung erforderlich.

4.3.1 Kodierung

Den Entwurf eines Genotyps zu einem gegebenen Phänotyp bzw. die Repräsentation dieses Genotyps im Rechner bezeichnet man als *Kodierung*. Die Wahl der Kodierung ist entscheidend für die Güte des Algorithmus.[4]

In der Natur können Chromosomensätze einfach (*haploid*), doppelt (*diploid*) oder mehrfach (*polyploid*) auftreten. Die meisten Kodierungen in der evolutionären Optimierung entsprechen haploiden Chromosomensätzen. Die Verwendung polyploider Chromosomensätze erleichtert neutrale Mutationen (siehe Abschnitt 4.3.2).

4.3.2 Neutrale Evolution

Unterscheidet man zwischen Genotyp und Phänotyp, so existieren zu einem Phänotyp meist mehrere Genotypen, die diesem zugeordnet sind. Bei einer statischen Fitnessfunktion haben diese die gleiche Fitness. Fasst man diejenigen Genotypen gleicher Fitness, die durch eine Mutation ineinander überführbar sind, als miteinander verbundene Knoten eines Graphen auf, erhält man ein neutrales Netzwerk von Genotypen.

Während ein lokales Suchverfahren bei identischem Genotyp und Phänotyp ein lokales Optimum nicht mehr verlassen kann, bieten neutrale Netzwerke die Möglichkeit, bei gleichbleibendem Fitnesswert den Genotyp zu verändern und so zu höheren Fitnesswerten zu gelangen, ohne zwischendurch niedrigere Fitnesswerte in Kauf zu nehmen.

Die Existenz von mehreren Genotypen zu einem Phänotyp lässt sich als eine Form von Redundanz auffassen. Untersuchungen [53] zum Einfluss dieser Redundanz auf den Verlauf evolutionärer Optimierungsverfahren haben ergeben, dass Redundanz den Verlauf der Optimierung positiv beeinflussen kann, aber nicht muss. Eine wichtige Voraussetzung für die positive Beeinflussung des Berechnungsverlaufs ist die Existenz großer neutraler Netzwerke.

4.3.3 Diversität

Unter der *Diversität* einer Population versteht man das Ausmaß der Heterogenität der in ihr enthaltenen Genotypen. Es gibt verschiedene Maße für Diversität innerhalb einer Population, z. B. die Anzahl vorhandener Werte für jedes Gen oder die Summe der paarweisen Entfernungen zwischen allen Individuen gemäß einem Distanzmaß auf den Genotypen.

Ausreichende Diversität ist eine Voraussetzung für den sinnvollen Einsatz von (kontrahierenden) Rekombinationsoperatoren, da diese bei sich ähnelnden Eltern zu diesen ähnliche Kinder erzeugen und somit im selben Teil des Suchraums bleiben. Können auch die Mutationsoperatoren nicht genug Diversität einführen (z. B. weil die Umweltselektion keine Verschlechterungen zulässt), bleibt der Algorithmus in einem lokalen Optimum gefangen. Es gibt mehrere Möglichkeiten, die Diversität einer Population zu erhalten oder wieder zu erhöhen:

Makromutationen Diese bewirken große Veränderungen im Genotyp und können dadurch weiter entfernte Teile des Suchraums erreichen. Lässt die Umweltselektion keine Verschlechterungen zu, kann man die Individuen nach der Mutation einem lokalen Optimierungsverfahren unterwerfen.

Immigration Unter Immigration (lat. immigrare, einwandern) verstehen wir die Einführung (zufälliger) neu erzeugter Individuen. Die Kontrolle über Häufigkeit und Umfang der Immigration kann über ein Maß für die Diversität der Population erfolgen.

[4] Kodierungen, bei denen kleine Änderungen im Genotyp große Änderungen im Phänotyp bewirken, können die Anzahl lokaler Optima stark erhöhen. Oft handelt es sich um Kodierungen in der Form von Zeichenketten über einem Alphabet kleiner Kardinalität; ein Beispiel ist die Kodierung einer natürlichen Zahl über dem Alphabet {0,1}. [48]

Nischenbildung ist die Separation von Individuen. Man unterscheidet verschiedene Techniken:

Regionale Populationsmodelle (auch Migrationsmodelle, Inselmodelle oder grobkörnige Modelle, engl. coarse grained models) unterteilen die Population in Teilpopulationen. Rekombination findet nur innerhalb der Teilpopulationen statt. Der Informationsaustausch zwischen den Teilpopulationen erfolgt über die Migration von Individuen.

Lokale Populationsmodelle (auch Diffusionsmodelle oder feinkörnige Modelle, engl. fine grained models) definieren für jedes Individuum eine Nachbarschaft. Individuen können nur innerhalb einer Nachbarschaft rekombiniert werden. Die Struktur der Nachbarschaftsbeziehung bestimmt, wie schnell sich Information durch die Population bewegen (diffundieren) kann.

Geteilte Fitnessfunktion Der Ansatz der geteilten Fitnessfunktion (engl. sharing function) benötigt ein Ähnlichkeitsmaß auf den Individuen (Genotyp oder Phänotyp). Die Fitness von Individuen, die vielen anderen Individuen der Population ähnlich sind, wird reduziert. Dadurch entsteht ein Selektionsdruck in Richtung Diversität der Population.

Verdrängung (engl. crowding) Dieser Ansatz erfordert ebenfalls ein Ähnlichkeitsmaß auf den Individuen. Ein durch Rekombination entstandenes Individuum ersetzt das ihm ähnlichste aus einer Auswahl von Individuen, etwa aus einer konstanten Anzahl von zufällig gezogenen Individuen oder seinen Eltern (deterministische Verdrängung).

Redundanz Die Verwendung polyploider Chromosomensätze. Meist wird in Anlehnung an die Natur ein dominanzbasierter Ansatz gewählt, bei welchem den möglichen Werten eines Gens (einer Position der Kodierung) die Eigenschaften dominant oder rezessiv zugewiesen werden. Bei Auswahlen, etwa der Rekombination, werden dann die dominanten Werte den rezessiven vorgezogen. Diese Methode eignet sich für Kodierungen ohne starke Abhängigkeiten der einzelnen Positionen untereinander.

Eine Alternative zum dominanzbasierten Ansatz ist der Pseudo-Meiose-Ansatz[5] [55]. Dieser verwendet ein Primärchromosom (engl. chief chromosome) und ein Assistenzchromosom (engl. assistant chromosome), denen unterschiedliche Funktionen zugeordnet sind. Für Rekombination und Mutation wird aus Primär- und Assistenzchromosom jedes Individuums durch interne Rekombination ein einfacher Chromosomensatz erstellt; die resultierenden Chromosomensätze ersetzen jeweils das Primärchromosom der Eltern. Die Assistenzchromosomen evolvieren unabhängig davon nur durch Mutation. Man kann den Vorgang als lokale Suche (durch die Evolution der Primärchromosomen) um die Assistenzchromosomen interpretieren. Die Methode eignet sich auch für dynamische (auch nicht stationäre) Probleme.

4.3.4 Nebenbedingungen

Im Folgenden nennen wir Elemente des nicht durch Nebenbedingungen eingeschränkten Lösungsraums, welche die Nebenbedingungen verletzen, *illegale* Lösungen. Es existieren unterschiedliche Ansätze zum Umgang mit Nebenbedingungen:

Bestrafung durch die Fitnessfunktion. Bei diesem Ansatz wird die Fitness von Individuen gemäß dem Ausmaß der Verletzung der Nebenbedingungen reduziert.

Ein Problem ist die Integration der Bestrafung in die Fitnessfunktion: Gibt man der Bestrafung einen hohen Stellenwert, läuft man bei stark restringierten Problemen Gefahr, dass das zuerst entstehende legale Individuum die restlichen, illegalen Individuen verdrängt und

[5] Bei der Meiose (auch Reifeteilung) von Zellen entstehen aus einer diploiden Zelle vier haploide Zellen. Insbesondere wird dabei ein diploider Chromosomensatz auf einen haploiden Chromosomensatz reduziert.

die Population so vorzeitig konvergiert. Gibt man der Bestrafung einen niedrigen Stellenwert, riskiert man, illegale Lösungen als Ergebnis zu erhalten, da sich durch die Verletzung der Nebenbedingungen die eigentliche Fitnessfunktion evtl. deutlich verbessern lässt. Dem Problem lässt sich entgegenwirken, indem man den Stellenwert der Bestrafung von der verstrichenen Zeit abhängig macht: Verletzungen der Nebenbedingungen werden zuerst nicht, mit zunehmender Generationenzahl dann immer schwerer bestraft.

Reparaturalgorithmen transformieren illegale Lösungen in legale Lösungen. Sie werden nach der Erzeugung bzw. der Veränderung eines Individuums ausgeführt und sind inhärent problemspezifisch.

Legale Operatoren Alle Methoden zur Erzeugung und Veränderung von Individuen, insbesondere Rekombinations- und Mutationsoperatoren, werden so entworfen, dass sie ausschließlich legale Lösungen liefern.

4.4 Bausteine für einen evolutionären Ansatz

In den folgenden Abschnitten beschreiben wir Algorithmen für die einzelnen Bestandteile — Kodierung, Populationsgröße, Initialisierung, Fitnessfunktion, Terminierungsbedingung, Paarungsselektion, Umweltselektion, Rekombinationsoperatoren, Mutationsoperatoren — eines evolutionären Algorithmus für das WCP-A. Dabei stellen wir für viele Bestandteile mehrere Rechenvorschriften vor. Ziel ist es, ein flexibles Rahmenwerk zu schaffen, in dem die unterschiedlichen Algorithmen für die einzelnen Bestandteile einfach ausgetauscht und kombiniert werden können („Baukastensystem“). Zu diesem Zweck stellen wir eine Methode vor, die eine einfache und flexible Kontrolle von Kombinationen mehrerer Algorithmen ermöglicht.

Die einzelnen Bauteile des Algorithmus können entweder *statisch* (zeitunabhängig) oder *dynamisch* (zeitabhängig) sein; dabei wird die verstrichene Zeit in der Anzahl der simulierten Generationen gemessen.

Im Folgenden sei $\mu \in \mathbb{N}$ die Größe der Population (Anzahl der Individuen zu Beginn einer Generation) und $\lambda \in \mathbb{N}$ die Anzahl der Kinder (Anzahl der in einer Generation neu erzeugten Individuen).[6] Wir bezeichnen mit $\tau - 1 \in \mathbb{N}_0$ die Anzahl der bereits simulierten Generationen. Eine Generation beginnt unmittelbar vor der Überprüfung der Terminierungsbedingung; die zu Beginn des Algorithmus neu initialisierte Population zählt als 0-te Generation. Mit f bezeichnen wir die Fitnessfunktion (siehe Abschnitt 4.4.6), mit $f_{\max}^{(i)}$ den besten Fitnesswert in Generation i, $0 \leq i < \tau$ und mit $f_{\text{best}}^{(k)} := \max\{f_{\max}^{(i)} | 0 \leq i \leq k\}$, $0 \leq k < \tau$ den besten Fitnesswert in allen Generationen bis zur k-ten Generation einschließlich.

4.4.1 Ablauf

Der Programmfluss des evolutionären Algorithmus entspricht dem Ablaufschema in Abbildung 4.1; dabei halten wir die einzelnen Bestandteile so eigenständig wie möglich. Sollen für einen Teil des Algorithmus mehrere Verfahren kombiniert werden, geschieht dies, soweit nicht anders vermerkt, mittels Algorithmus 4.2 aus Abschnitt 4.4.2. Algorithmus 4.1 beschreibt den Verlauf des evolutionären Algorithmus im Detail.

[6] Den Fall $\lambda = 0$ schließen wir aus, da weder Rekombination noch Mutation erfolgen würden. Möchte man nur ein einzelnes Individuum verändern, z. B. um Verfahren wie das simulierte Abkühlen oder die iterierte lokale Suche abzubilden, kann man dies durch $\mu = \lambda = 1$ und eine (μ, λ)-Umweltselektion (siehe Abschnitt 4.4.8.2) erreichen.

Seien $\mu, \lambda \in \mathbb{N}$ gegeben.

(1) Initialisierung.

(a) Erzeuge μ neue Individuen $I_1, \ldots, I_\mu$ (Abschnitt 4.4.5).

(b) Setze $\tau \leftarrow 1$, $f_{\text{best}}^{(0)} \leftarrow f_{\text{max}}^{(0)} \leftarrow \max\{f(I_i) \mid 1 \leq i \leq \mu\}$.

(2) Solange die Terminierungsbedingung (Abschnitt 4.4.7) nicht erfüllt ist, wiederhole:

(a) Führe Algorithmus 4.5 zur Paarungsselektion (Abschnitt 4.4.8.1) durch.

(b) Erzeuge λ Individuen durch Rekombination (Abschnitt 4.4.9).

(c) Mutiere jedes dieser Individuen (Abschnitt 4.4.10).

(d) Wähle μ Individuen $I_1 \ldots, I_\mu$ aus (Umweltselektion, Abschnitt 4.4.8.2).

(e) Setze $f_{\text{max}}^{(\tau)} \leftarrow \max\{f(I_i) \mid 1 \leq i \leq \mu\}$, $f_{\text{best}}^{(\tau)} \leftarrow \max\left\{f_{\text{max}}^{(\tau)}, f_{\text{best}}^{(\tau-1)}\right\}$.

(f) Setze $\tau \leftarrow \tau + 1$.

Zu Beginn von Schritt 2a und Schritt 2d müssen die Individuen nach ihrer Fitness in aufsteigender Reihenfolge sortiert vorliegen.

ALGORITHMUS 4.1: Ein evolutionärer Algorithmus für das WCP-A

4.4.2 Kombination mehrerer Algorithmen

Sollen in Algorithmus 4.1 an einer der Stellen 1a, 2b, 2c oder 2d mehrere Methoden zum Einsatz kommen, z. B. verschiedene Algorithmen zur Konstruktion von Startlösungen oder verschiedene Mutationsoperatoren, muss geregelt werden, welche der Methoden in welcher Reihenfolge und wie oft auszuführen sind. Wir wenden dazu einheitlich Algorithmus 4.2 an.

Gegeben seien $p \in \mathbb{N}_0$ Einheiten einer Ressource und $q \in \mathbb{N}$ Verbraucher, wobei jedem Verbraucher eine absolute Anforderung $a_i \in \mathbb{N}_0$ und eine relative Anforderung $r_i \in \mathbb{N}_0$, $1 \leq i \leq q$ zugeordnet sind. Es gelte $\sum_{i=1}^{q} a_i \leq p$ und $\sum_{i=1}^{q} r_i > 0$. Die Ressourceneinheiten werden den Verbrauchern wie folgt zugeordnet:

(1) Erfülle die absoluten Anforderungen, d. h. weise dem i-ten Verbraucher a_i Einheiten zu. Es sei $p' \leftarrow p - \sum_{i=1}^{q} a_i$ die Anzahl der noch zu verteilenden Einheiten.

(2) Sei $r'_i \leftarrow r_i / \sum_{j=1}^{q} r_j$, $1 \leq i \leq q$ der (prozentuelle) Anteil der i-ten relativen Anforderung an allen relativen Anforderungen. Weise dem i-ten Verbraucher $\lfloor r'_i \cdot p' \rfloor$ Einheiten zu.

(3) Verteile die restlichen $p' - \sum_{i=1}^{q} \lfloor r'_i \cdot p' \rfloor$ Einheiten gemäß der durch die Wahrscheinlichkeiten r'_i definierten Verteilung auf die Verbraucher.

ALGORITHMUS 4.2: Ressourcenverteilung bei absoluten und relativen Anforderungen

In Schritt 1 werden zuerst die absoluten und in Schritt 2, soweit ganzzahlig möglich, die relativen Anforderungen erfüllt. Die übrigen Einheiten werden entsprechend den verbleibenden Restanforderungen stochastisch verteilt. Aufgrund der Voraussetzung $\sum_{i=1}^{q} a_i \leq p$ ist Schritt 1 stets möglich. In Schritt 2 kann es wegen $\sum_{i=1}^{q} r_i > 0$ nicht zu einer Division durch Null kommen. Die r'_i können wegen $0 \leq r'_i \leq 1$ und $\sum_{i=1}^{q} r'_i = 1$ als Wahrscheinlichkeiten interpretiert werden. Man beachte,

dass im Fall $p' = 0$ in den Schritten 2 und 3 keine Einheiten verteilt werden. Das Verfahren, insbesondere Schritt 3, stellt sicher, dass die Verteilung der Ressourceneinheiten bei häufiger Ausführung des Verfahrens exakt den Anforderungen entspricht.[7]

Beispiel 4.1 (Initialisierung einer Population) Für einen evolutionären Algorithmus seien zur Initialisierung der Population $p = 4$ Individuen zu konstruieren; dabei stehen $q = 3$ Methoden zur Verfügung. Die erste, eine deterministische Heuristik, soll genau einmal ausgeführt werden. Die beiden anderen Methoden erzeugen, auf zwei unterschiedliche Arten, zufällige Individuen; sie sollen gleich oft ausgeführt werden. Wir setzen dazu $a_1 = 1$, $a_2 = a_3 = 0$, $r_1 = 0$, $r_2 = r_3 = 1$.

In Schritt 1 wird der ersten Methode eine Einheit zugewiesen. Es verbleiben $p' = 3$ Einheiten.

In Schritt 2 werden der ersten Methode $\lfloor 0 \cdot 3 \rfloor = 0$ Einheiten, den beiden anderen Methoden jeweils $\lfloor \frac{1}{2} \cdot 3 \rfloor = 1$ Einheit zugewiesen.

In Schritt 3 wird die restliche $3 - 2 = 1$ Einheit mit einer Wahrscheinlichkeit von $r'_1 = 0$ der ersten Methode und mit einer Wahrscheinlichkeit von jeweils $r'_2 = r'_3 = \frac{1}{2}$ einer der beiden anderen Methoden zugewiesen. □

Das vorgestellte Verteilungsverfahren hat den Vorteil, dass es eine einfache Schnittstelle zur einheitlichen Manipulation aller Bestandteile des evolutionären Algorithmus zur Verfügung stellt. Die Anforderungen sind leicht zu überprüfen und zu erfüllen.

Sind die Ressourceneinheiten auf die Verbraucher verteilt worden, können diese sie entweder *blockweise* oder *schrittweise* verbrauchen. Bei blockweisem Verbrauch konsumiert nacheinander jeder Verbraucher die ihm zugewiesenen Einheiten am Stück. Bei schrittweisem Verbrauch konsumieren die Verbraucher, sofern sie noch über zugewiesene Einheiten verfügen, reihum jeweils eine Einheit; dies wird so lange fortgesetzt, bis alle Verbraucher die ihnen zugewiesenen Einheiten konsumiert haben.

4.4.3 Kodierung

Wir verzichten auf eine Trennung von Genotyp und Phänotyp, d. h. der Algorithmus arbeitet direkt mit der Menge aller Lösungskandidaten für das WCP-A als Suchraum. Unter Lösungskandidaten verstehen wir dabei Objekte, die den syntaktischen[8] Anforderungen aus Definition 3.1 entsprechen, d. h. angeordnete Partitionierungen von V mit $\sigma_i(1) = s_i$. Wir entwickeln eine Kodierung für die Lösungskandidaten mit den Zielen einer guten Implementierbarkeit und der Unterstützung von Teillösungen.

Ein Lösungskandidat für das WCP-A besteht aus einer Partitionierung von V in m Mengen; jede dieser Mengen ist durch eine bijektive Funktion $\sigma_i : \{1, \ldots, |L_i|\} \to L_i$ angeordnet und es gilt $\sigma_i(1) = m_i$. Da der Startknoten jeder Partition feststeht, müssen die m_i nicht explizit gespeichert werden, und es genügt, die restlichen Knoten zu betrachten. O. B. d. A. sei $V \setminus \{s_1, \ldots, s_m\} = \{1, \ldots, n\}$.

Wir verwenden als Kodierung ein zweidimensionales Feld von ganzen Zahlen; der erste Index entspricht der Partition, der zweite Index entspricht der durch die Anordnung vorgegebenen Reihenfolge innerhalb dieser Partition. Durch die Kodierung der Reihenfolge in den Index des Feldes entfällt die Notwendigkeit einer expliziten Speicherung der σ_i. Wir führen die folgende Notation für Lösungskandidaten ein:

[7] Läßt man $p, q, a_1, \ldots, a_q, r_1, \ldots, r_q$ fest, führt das Verteilungsverfahren mehrmals aus und bildet für jeden Verbraucher den Durchschnitt über die auf ihn verteilten Ressourceneinheiten, so ergeben sich im Grenzwert (Anzahl der Ausführungen des Verfahrens gegen Unendlich) exakt die angeforderten $a_i + r'_i \cdot p'$ Einheiten.

[8] Gemeint sind die Anforderungen an die formale Struktur der Objekte. Lösungskandidaten unterscheiden sich von Lösungen dadurch, dass sie nicht optimal sein müssen.

Definition 4.1 (Individuum) Es sei

$$\mathcal{F} := \left\{ \left\{ \{1,\ldots,n\}^k \;\middle|\; 0 \le k \le n \right\}^m \;\middle|\; \left| \bigcup_{i=1}^{m} \bigcup_{j=1}^{|L_i|} \{L_{i,j}\} \right| = \sum_{i=1}^{m} |L_i| = n \right\}. \tag{4.1}$$

Wir nennen $L \in \mathcal{F}$ ein *Individuum* (oder Lösungskandidaten) und $\mathcal{F}$ die Menge aller Individuen. Ist $\left|\bigcup_{i=1}^{m} \bigcup_{j=1}^{|L_i|} \{L_{i,j}\}\right| = \sum_{i=1}^{m} |L_i| < n$ sprechen wir von einer *partiellen Lösung*. Besteht keine Verwechslungsgefahr, schreiben wir kurz L_i für $(L)_i$, $L_{i,j}$ für $\big((L)_i\big)_j$ und $L_{i,j\ldots k}$ für $L_{i,j}, L_{i,j+1}, \ldots, L_{i,k}$. □

Die Bedingung in Gleichung 4.1 stellt sicher, dass jeder Knoten genau einmal in L enthalten ist.[9] Bei der Notation ist Folgendes zu beachten: Sind $I_1, \ldots, I_\mu$ Individuen und möchte man sich auf den k-ten Schweißpunkt der j-ten Partition des i-ten Individuums beziehen, geschieht dies durch $(I_i)_{j,k}$. Ist dagegen L ein Individuum, kann das Klammerpaar entfallen und es genügt $L_{j,k}$.

4.4.4 Populationsgröße und Anzahl der Nachkommen

Die Anzahl der über den gesamten Verlauf des Algorithmus hinweg berechneten Individuen hängt von der Größe μ der Population und der Anzahl λ der in jeder Generation erzeugten Nachkommen ab. Wir betrachten zwei statische Verfahren zur Angabe von μ und λ, absolute und relative Angaben.[10]

Bei absoluten Angaben wird μ und λ ein konstanter Wert zugewiesen. Bei relativen Angaben errechnet sich der Wert für μ bzw. λ aus dem Produkt einer Basisgröße und eines Faktors. Dabei bezieht sich der Faktor $\mu' \in \mathbb{R}_{>0}$ für die Berechnung von μ auf das Maß $n+m$ für die Eingabegröße und der Faktor $\lambda' \in \mathbb{R}_{>0}$ für die Berechnung von λ auf die Populationsgröße μ:

$$\mu = \lceil (n+m) \cdot \mu' \rceil \quad \text{und} \quad \lambda = \lceil \mu \cdot \lambda' \rceil.$$

Die Verwendung der oberen Gaußklammer stellt $\mu > 0$ und $\lambda > 0$ sicher. Man beachte, dass Kombinationen von absoluten und relativen Angaben zulässig sind. Abhängig von der Art der Umweltselektion (siehe Abschnitt 4.4.8.2) gibt es Einschränkungen für die Werte von λ und λ':

Bei der Komma-Selektion (μ, λ) muss $\lambda \ge \mu$ gelten; für die Spezifikation von λ durch relative Angaben gilt die Einschränkung $\lambda' \ge 1$. Die gleichzeitige Spezifikation von μ durch eine relative und von λ durch eine absolute Angabe ist unzulässig, da sich bei genügend großem $n+m$ immer $\lambda < \mu$ ergibt. Bei der Plus-Selektion $(\mu + \lambda)$ gibt es keine Einschränkungen, da wegen $\lambda > 0 \Rightarrow \mu + \lambda > \mu$ immer genug Individuen zur Verfügung stehen und mindestens ein neues Individuum erzeugt wird. Tabelle 4.2 fasst alle Einschränkungen zusammen.

Tabelle 4.2: Durch die Art der Umweltselektion bedingte Einschränkungen bei den Angaben zu Populationsgröße und Anzahl der Nachkommen

Konstellation	Einschränkung (μ, λ)	Einschränkung $(\mu + \lambda)$
μ absolut, λ absolut	$\lambda \ge \mu$	keine
μ absolut, λ relativ	$\lambda' \ge 1$	keine
μ relativ, λ absolut	unzulässig	keine
μ relativ, λ relativ	$\lambda' \ge 1$	keine

[9] Die Anforderung an die Summe der Partitionslängen stellt sicher, dass L genau n Knoten enthält, während die Anforderung an die Kardinalität der Vereinigungsmenge gewährleistet, dass diese verschieden sind. Da nur die Knoten $\{1, \ldots, n\}$ zur Verfügung stehen, enthält L jeden dieser Knoten genau einmal.

[10] Nicht zu verwechseln mit den absoluten und relativen Anforderungen aus Abschnitt 4.4.2 zur Kombination mehrerer Algorithmen.

4.4.5 Initialisierung der Population

Wir unterscheiden zwei Ansätze zur Erzeugung von Individuen: Die zufällige Erzeugung von Individuen, bei der Abdeckung und Erreichbarkeit des Suchraums im Vordergrund stehen, sowie Heuristiken zur Konstruktion guter Startlösungen. Wir stellen jeweils einen Algorithmus vor.

4.4.5.1 Uniforme Verteilung

Ziel ist es, aus allen möglichen Individuen eines zufällig zu bestimmen; dabei soll jedes Individuum die gleiche Wahrscheinlichkeit besitzen. Wir bringen dazu die Knoten des Graphen in eine zufällige Reihenfolge und verteilen anschließend zufällig $m-1$ Trennstellen. Algorithmus 4.3 beschreibt das Verfahren im Detail; Beispiel 4.2 verdeutlicht exemplarisch den Ablauf. Es gilt

SATZ 4.1 Algorithmus 4.3 definiert eine uniforme Verteilung auf allen Individuen. □

BEWEIS Wir zeigen zuerst, dass jedes mögliche Individuum durch den Algorithmus erzeugt werden kann. Sei L ein Individuum. Dann wird durch

$$L_{1,1}, L_{1,2}, \ldots, L_{1,|L_1|}, L_{2,1}, L_{2,2}, \ldots, L_{2,|L_2|}, \ldots, L_{m,1}, L_{m,2}, \ldots, L_{m,|L_m|}$$

eine Permutation σ auf $1, \ldots, n$ definiert. Mit σ und $t_k := 1 + \sum_{i=1}^{k} |L_i|$, $1 \leq k < m$ liefert Algorithmus 4.3 gerade L als Ergebnis.

Die Wahrscheinlichkeit von L, durch den Algorithmus erzeugt zu werden, ist das Produkt der Wahrscheinlichkeit für die Wahl von σ und der Wahrscheinlichkeit für die Wahl von $t_1, \ldots, t_{m-1}$. Die erste Wahrscheinlichkeit ist das Inverse der Anzahl der Permutationen der Länge n, die zweite Wahrscheinlichkeit ist das Inverse der Anzahl der Möglichkeiten, $m-1$ Elemente mit Wiederholungen, aber ohne Reihenfolge aus $n+1$ Elementen zu ziehen (Ziehung einer $(m-1)$-elementigen Multimenge aus $n+1$ Elementen). Als Gesamtwert ergibt sich gerade das Inverse der Größe des Suchraums:

$$\mathbb{P}[L] = \frac{1}{n!} \frac{1}{\left(\!\binom{n+1}{m-1}\!\right)} = \frac{1}{n!} \frac{1}{\binom{(n+1)+(m-1)-1}{m-1}} = \frac{1}{n!\binom{n+m-1}{m-1}} = \frac{1}{n!\binom{n+m-1}{n}}.$$

□

Seien $\{1, \ldots, n, s_1, \ldots, s_m\}$ die Knoten des Graphen.

(1) Bestimme uniform verteilt eine zufällige Permutation σ der Länge n.[a]

(2) Bestimme uniform verteilt und mit Wiederholungen $m-1$ Trennstellen $t_1, \ldots, t_{m-1} \in \{1, \ldots, n+1\}$. Sortiere diese aufsteigend.

(3) Bezeichne $t'_1 \leq \cdots \leq t'_{m-1}$ die sortierten Trennstellen. Setze $t'_0 \leftarrow 1$, $t'_m \leftarrow n+1$ und

$$L_i \leftarrow \sigma(t'_{i-1}), \sigma(t'_{i-1}+1), \ldots, \sigma(t'_i - 1), \quad 1 \leq i \leq m.$$

Es gilt $|L_i| = t'_i - t'_{i-1}$; im Fall $t'_{i-1} = t'_i$ besteht L_i nur aus der Startposition s_i (die nicht kodiert wird). Für $m = 1$ wird keine Trennstelle bestimmt und L_1 ist die durch σ bestimmte Tour.

[a] Dies kann z. B. in n Schritten durch uniforme Auswahl des jeweils nächsten Elements der Permutation aus den verbleibenden Elementen geschehen.

ALGORITHMUS 4.3: Uniforme Erzeugung von Lösungskandidaten

Beispiel 4.2 (Uniforme Erzeugung einer Lösung für das WCP-A) Sei $n = 4$ und $m = 3$. In Schritt 1 ziehen wir für die erste Position von σ gleichverteilt ein Element aus {1,2,3,4}. Sei 3 das Ergebnis der Ziehung. Für die zweite Position von σ ziehen wir gleichverteilt aus {1,2,4} mit dem Ergebnis 4, für die dritte Position 1 aus {1,2}. Wir erhalten die Permutation σ mit $\sigma\big((1,2,3,4)\big) = (3,4,1,2)$.

In Schritt 2 ziehen wir zwei Trennstellen, $t_1, t_2 \in \{1,2,3,4,5\}$ mit dem Ergebnis $t_1 = 5$ und $t_2 = 3$. Sortiert ergibt sich $t'_1 = 3$ und $t'_2 = 5$.

In Schritt 3 setzen wir $t'_0 := 1$ sowie $t'_3 := 5$ und erhalten als Lösung

$$\begin{aligned} L_1 &= s_1, \sigma(1), \sigma(2), \ldots, \sigma(2) = s_1,3,4 \\ L_2 &= s_2, \sigma(3), \sigma(4), \ldots, \sigma(4) = s_2,1,2 \\ L_3 &= s_3, \sigma(5), \sigma(6), \ldots, \sigma(4) = s_3. \end{aligned}$$

□

4.4.5.2 Nächster Roboter & Gierige Ersparnis-Heuristik

Wir geben eine Heuristik zur Konstruktion einer guten Startlösung. Die Heuristik besteht aus zwei Teilen: Zuerst werden die Schweißpunkte auf die Roboter verteilt. Anschließend werden die Schweißpunkte jedes Roboters durch die gierige Ersparnis-Heuristik (engl. greedy savings heuristic) [71] für das TSP angeordnet.

Die Verteilung der Schweißpunkte auf die Roboter erfolgt anhand des Abstands zwischen Schweißpunkt und Roboter. Sie kann entweder deterministisch durch Zuordnung zum nächsten Roboter erfolgen oder stochastisch durchgeführt werden, indem der Anteil des jeweiligen Abstands an der Summe der Abstände zu allen Startknoten als Wahrscheinlichkeit interpretiert wird.

Da wir gemäß Definition 3.1 auf einem vollständigen Graphen arbeiten, genügt zur Bestimmung des nächsten Roboters die Betrachtung der Kanten zwischen einem Schweißpunkt und den Startknoten, falls die Kantengewichtung die Dreiecksungleichung erfüllt. Andernfalls können die nächsten Roboter durch einen kürzeste Wege Algorithmus für alle Paare (engl. all pairs shortest path problem) wie den Floyd-Warshall-Algorithmus [80] gefunden werden. Alternativ kann auch Dijkstras Algorithmus [80] m-mal ausgeführt werden.

Die gierige Heuristik für das TSP basiert auf folgender Idee: Die Kanten werden ihrem Gewicht nach aufsteigend sortiert. Anschließend geht man die Kanten durch und nimmt alle diejenigen Kanten in die entstehende Tour auf, bei deren Hinzunahme die bisher aufgenommenen Kanten noch zu einer Tour auf dem ganzen Graphen fortgesetzt werden können. Algorithmus 4.4 beschreibt das Verfahren im Detail.

Beispiel 4.3 (Nächster Roboter & Gierige Ersparnis-Heuristik) Sei $m = 2$ und $n = 3$. Wir verwenden den euklidischen Abstand in der Ebene als Kantengewichtung. Abbildung 4.2 zeigt die Station und die Kantengewichte des entsprechenden Graphen.

In Schritt 1 werden die Knoten 1,2,3 dem Startknoten s_1 zugewiesen, da sie diesem am nächsten sind:

$$\begin{aligned} \arg\min \big\{ c\big(\{1, s_j\}\big) \;\big|\; 1 \le j \le 2 \big\} &= 1 \\ \arg\min \big\{ c\big(\{2, s_j\}\big) \;\big|\; 1 \le j \le 2 \big\} &= 1 \\ \arg\min \big\{ c\big(\{3, s_j\}\big) \;\big|\; 1 \le j \le 2 \big\} &= 1 \quad . \end{aligned}$$

Somit ist $L_1 = \{s_1,1,2,3\}$ und $L_2 = \{s_2\}$. Aufgrund von $|L_1| = 4 > 2$ führen wir für L_1 die Schritte 2a bis 2c aus. Zuerst sortieren wir alle Kanten in L_1 nach ihren Kosten:

$$\begin{array}{cccccccccccc} & c\big(\{1,2\}\big) & \le & c\big(\{2,3\}\big) & \le & c\big(\{1,3\}\big) & \le & c\big(\{1,s_1\}\big) & \le & c\big(\{3,s_1\}\big) & \le & c\big(\{2,s_1\}\big) \\ \iff & 1{,}414 & \le & 1{,}414 & \le & 2 & \le & 2{,}236 & \le & 2{,}236 & \le & 3 \end{array} .$$

(1) Ordne jeden Schweißpunkt seiner nächsten Startposition zu. Bei Uneindeutigkeiten bevorzuge zuerst die Partition mit den wenigsten Knoten und dann diejenige mit dem kleinsten Index.

Erfüllt die Kantengewichtung die Dreiecksungleichung, kann dies für den Knoten i durch Bestimmung von $\arg\min\{c(\{i,s_j\}) \mid 1 \leq j \leq m\}$ geschehen. Andernfalls müssen kürzeste Wege bestimmt werden, z. B. durch den Floyd-Warshall-Algorithmus oder mehrfache Ausführung von Dijkstras Algorithmus.

(2) Sei L_i die Knotenmenge, die s_i in Schritt 1 zugeordnet wurde, einschließlich s_i selbst.
Führe für alle L_i mit $|L_i| > 2$ die folgenden Schritte durch:

(a) Sortiere alle Kanten mit beiden Endpunkten in L_i in aufsteigender Reihenfolge nach ihren Kosten. Bezeichne $c_1 \leq c_2 \leq \cdots \leq c_{|L_i|^2}$ die sortierten Kanten.

(b) Setze $T \leftarrow \emptyset$, $j \leftarrow 1$.

(c) Solange $|T| < |L_i|$ wiederhole:

(i) Falls beide Knoten von c_j Grad 0 oder 1 bezüglich T haben und es zwischen den beiden Knoten keinen Weg in T gibt (außer $T \cup \{c_j\}$ ist ein Hamiltonkreis), setze $T \leftarrow T \cup \{c_j\}$.

(ii) Setze $j \leftarrow j + 1$.

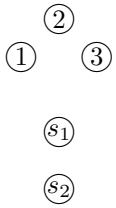
Die Überprüfung der Bedingung in Zeile 2ci kann mit Hilfe der üblichen Datenstruktur für disjunkte Mengen [80] praktisch in Konstantzeit erfolgen. Die Bedingung bezüglich des Hamiltonkreises in Schritt 2ci ist nur für die letzte hinzuzunehmende Kante von Bedeutung.

Schritt 1 kann alternativ stochastisch ausgeführt werden:

(1) Ordne jeden Schweißpunkt gemäß den folgenden Wahrscheinlichkeiten einer Startposition zu:

$$\mathbb{P}[j \in L_i] := \frac{c(\{j,m_i\})}{\sum_{k=1}^{m} c(\{j,m_k\})}.$$

ALGORITHMUS 4.4: Deterministische und stochastische Ausführung der nächster Roboter & gierige Ersparnis-Heuristik zur Erzeugung von Lösungskandidaten für das WCP-A



(a) Station

	Knoten				
Knoten	1	2	3	s_1	s_2
1		1,414	2,000	2,236	3,162
2			1,414	3,000	4,000
3				2,236	3,162
s_1					1,000
s_2					

(b) Kantengewichte

Abbildung 4.2: Beispiel für die nächster Roboter & gierige Ersparnis-Heuristik

Wir beginnen mit $T = \emptyset$ und $j = 1$. Da $|T| = 0 < 4 = |L_i|$ fahren wir mit Schritt 2ci fort: Beide Knoten von $c_1 = \{1, 2\}$ haben Grad 0 in T, und da T keine Kanten enthält, gibt es auch keinen Weg zwischen 1 und 2 in T. Wir setzen $T \leftarrow T \cup \big\{\{1, 2\}\big\} = \big\{\{1, 2\}\big\}$ und erhöhen j auf 2.

Da $|T| = 1 < 4 = |L_i|$ betrachten wir $c_2 = \{2, 3\}$. Knoten 2 hat Grad 1 in T, Knoten 3 hat Grad 0 in T und die Knoten 2 und 3 sind nicht durch Kanten aus T miteinander verbunden. Wir nehmen daher $\{2, 3\}$ in T auf, erhöhen j auf 3 und fahren wegen $|T| = 2 < 4 = |L_i|$ mit Schritt 2ci fort.

Die Knoten von Kante $c_3 = \{1, 3\}$ haben beide Grad 1 in T, jedoch gibt es einen Weg (der durch die Hinzunahme von c_3 nicht zu einem Hamiltonkreis wird) über die Kanten $\{1, 2\}$ und $\{2, 3\}$ von 1 nach 3 in T. Wir lassen daher T unverändert und erhöhen j auf 4.

Wie gehabt überprüfen wir die Kante $c_4 = \{1, s_1\}$, nehmen diese in T auf und erhöhen j auf 5. Danach ist $|T| = 3 < 4 = |L_i|$ (es fehlt noch die letzte Kante), und wir betrachten $c_5 = \{3, s_1\}$. Sowohl 3 als auch s_1 haben Grad 1 in T; zwar gibt es einen Weg von 3 nach s_1 in T, aber

$$T \cup \{c_5\} = \big\{\{1{,}2\}, \{2{,}3\}, \{1, s_1\}, \{3, s_1\}\big\}$$

ist ein Hamiltonpfad. Wir nehmen c_5 in T auf, erhöhen j auf 6 und terminieren wegen $|T| = |L_i|$ in Schritt 2. □

Ist die Anzahl der Roboter durch die Anzahl der Schweißpunkte beschränkt, gilt für die Laufzeit von Algorithmus 4.4

Satz 4.2 Algorithmus 4.4 hat für $m \leq n$ eine worst-case Laufzeit von $O\big(mn^2 \log^* n\big)$. □

Beweis In Schritt 1 sind $O(n \cdot m)$ Kantengewichte zu betrachten, falls diese der Dreiecksungleichung genügen. Ist dies nicht der Fall, müssen die Kosten kürzester Wege von den gewöhnlichen Knoten zu den Startknoten berechnet werden. Der Floyd-Warshall-Algorithmus hat eine worst-case Laufzeit von $O\big(|V|^3\big) = O\big((n + m)^3\big)$. [80] Alternativ kann man, da die Kantengewichte laut Definition 3.1 nicht negativ sind, Dijkstras Algorithmus für jeden Startknoten einmal ausführen. Da der Graph ungerichtet ist, erhält man die kürzesten Wegkosten von jedem Knoten aus zu jedem Startknoten. Die worst-case Laufzeit von Dijkstras Algorithmus ist $O\big(|V|^2\big)$ [80], es ergibt sich $m \cdot O\big(|V|^2\big) = O\big(m(n + m)^2\big)$. Für den von uns angenommenen Fall $m \ll n$ ist dies eine merkbar bessere Laufzeit.

Für Schritt 2 schätzen wir nach oben ab und nehmen an, dass die Schleife m-mal mit $|L_i| = n+1$ ausgeführt wird. Das Sortieren der Kanten erfordert $O\big(|L_i| \log(|L_i|)\big) = O\big((n + 1) \log(n + 1)\big) = O\big(n \log n\big)$ Rechenschritte. In Schritt 2ci ist die Bedingung auszuwerten. Der Grad der beiden

Knoten kann mit Hilfe eines entsprechenden Feldes (engl. array) in Konstantzeit ermittelt werden, ebenso, ob $T \cup \{c_j\}$ einen Hamiltonkreis bildet: Dies ist genau dann der Fall, wenn c_j die letzte einzufügende Kante ist und c_j zwei Knoten mit Grad 1 in T verbindet. Ohne Berücksichtigung der Kosten für die Überprüfung, ob es in T einen Weg zwischen den beiden Knoten von c_j gibt, fallen damit für Schritt 2c höchstens $|L_i|^2 = (n+1)^2$ mal konstante Kosten an.

Für die Überprüfung, ob es in T einen Weg zwischen den beiden Knoten von c_j gibt, führen wir eine amortisierte Analyse für den gesamten Schritt 2c durch. Mit Hilfe der üblichen Datenstruktur für disjunkte Mengen können x Operationen (Erstellen einer einelementigen Menge, Vereinigung zweier Mengen, Abfrage des Repräsentanten), von denen y Erstellen-Operationen sind, in Zeit $O\big(x \log^* y\big)$ erfolgen. [80] Die Vereinigung $T \leftarrow T \cup \{c_j\}$ wird höchstens $|L_i| = n+1$ mal ausgeführt. Es ergibt sich mit

$$y = |L_i| = n+1 \quad \text{und} \quad x = y + 1(n+1) + 2(n+1)^2 = 2(n^2+3n+2)$$

eine Laufzeit von $O\big((n^2+3n+2)\log^*(n+1)\big) = O\big(n^2 \log^* n\big)$. Da es sich bei $\log^*$ im Wesentlichen um das Inverse der Ackermann-Funktion handelt, ist $O\big(\log^* n\big)$ für praktische Zwecke konstant. Insgesamt ergibt sich für Schritt 2 eine Laufzeit von

$$m\left(O\big(n \log n\big) + O\big(n^2\big) + O\big(n^2 \log^* n\big)\right) = O\big(m\, n^2 \log^* n\big).$$

Gilt die Dreiecksungleichung für die Kantengewichte, ergibt sich eine worst-case Laufzeit des Algorithmus von $O\big(nm + mn^2 \log^* n\big) = O\big(mn^2 \log^* n\big)$, andernfalls von

$$O\big(m(n+m)^2 + mn^2 \log^* n\big) = O\big(mn^2 + 2m^2 n + m^3 + mn^2 \log^* n\big) \overset{m \leq n}{=} O(mn^2 \log^* n).$$

In beiden Fällen dominiert Schritt 2 die Laufzeit des Algorithmus. □

4.4.6 Fitnessfunktion

Die Fitness eines Individuums entspricht dem Wert der kodierten Lösung. Dieser wird mit einem negativen Vorzeichen versehen, da der Wert der Lösung zu minimieren, die Fitness hingegen zu maximieren ist:

$$f(L) := -\max\left\{ c\big(\{s_i, L_{i,1}\}\big) + c\big(\{s_i, L_{i,|L_i|}\}\big) + \sum_{j=1}^{|L_i|-1} c\big(\{L_{i,j}, L_{i,j+1}\}\big) \;\middle|\; 1 \leq i \leq m \right\}.$$

Der Aufwand zur Berechnung von f beläuft sich bei Verwendung einer Distanzmatrix auf $O(n+m)$.

Die Auswertung der Fitnessfunktion kann verzögert erfolgen (Auswertung bei Bedarf, engl. lazy evaluation). Dabei wird die Fitness eines Individuums nur dann neu berechnet, wenn sie benötigt wird (z. B. beim Vergleich der Fitnesswerte zweier Individuen) und sich das betroffene Individuum seit der letzten Berechnung seiner Fitness verändert hat (z. B. durch Initialisierung, Rekombination, Mutation). Der Einsatz der verzögerten Auswertung lohnt sich umso mehr, je aufwendiger die Berechnung der Fitnessfunktion ist. Er beeinflusst das Terminierungskriterium der maximalen Anzahl an Auswertungen der Fitnessfunktion (siehe Abschnitt 4.4.7).

Wir bezeichnen die Anzahl der (verzögerten) Auswertungen der Fitnessfunktion in der i-ten Generation, $0 <= i < \tau$, mit $f_{\text{eval}}^{(i)}$.

Die Werte der Fitnessfunktion sind, aufgrund der nicht negativen reellen Kantengewichte, nicht positive reelle Zahlen. Auf diesen ist durch die übliche $<$ Relation eine totale Ordnung gegeben.

4.4.7 Terminierungsbedingung

Unter einer Terminierungsbedingung verstehen wir eine Funktion mit Wertebereich $\{\text{wahr}, \text{falsch}\}$, der als Definitionsbereich die Werte τ sowie $f_{\max}^{(i)}$, $f_{\text{best}}^{(i)}$ und $f_{\text{eval}}^{(i)}$ für $0 \leq i < \tau$ zur Verfügung stehen. Sollen mehrere Terminierungsbedingungen gleichzeitig zum Einsatz kommen, geschieht dies durch Disjunktion: $\text{term}_1 \vee \text{term}_2 \vee \ldots \vee \text{term}_k$. Es muss mindestens eine Terminierungsbedingung angegeben werden. Wir stellen drei Terminierungsbedingungen vor:

Maximale Anzahl an Generationen Der Algorithmus terminiert sobald eine festgelegte Anzahl an Generationen simuliert wurde. Sei $\tau_{\max} \in \mathbb{N}_0$. Wir setzen

$$\text{term}_1(\tau) := \begin{cases} \text{wahr} & \text{falls } \tau - 1 \geq \tau_{\max}, \\ \text{falsch} & \text{sonst.} \end{cases}$$

Keine Verbesserung der besten Individuen Der Algorithmus terminiert, wenn die Fitness des jeweils besten Individuums einer Generation sich innerhalb der letzten $\tau_{\text{last}} \in \mathbb{N} \setminus \{1\}$ Generationen nicht verbessert hat:

$$\text{term}_2\big(\tau, f_{\max}^{(0)}, \ldots, f_{\max}^{(\tau-1)}\big) := \begin{cases} \text{falsch} & \text{falls } \tau < \tau_{\text{last}} \ \vee \ \exists i \in \{\tau - \tau_{\text{last}}, \ldots, \tau - 2\} : f_{\max}^{(i)} < f_{\max}^{(i+1)}, \\ \text{wahr} & \text{sonst.} \end{cases}$$

Das Terminierungskriterium hat stets den Wert falsch, wenn noch nicht genügend Generationen simuliert wurden, um τ_{last} Fitnesswerte überprüfen zu können. Man beachte, dass die in Algorithmus 4.1 in Schritt 1a initialisierte Population als 0-te Generation zählt. Alternativ kann term_2 über die bisher besten Fitnesswerte ausgedrückt werden:

$$\text{term}_2\big(\tau, f_{\text{best}}^{(0)}, \ldots, f_{\text{best}}^{(\tau-1)}\big) = \begin{cases} \text{falsch} & \text{falls } \tau < \tau_{\text{last}} \ \vee \ \left|\left\{ f_{\text{best}}^{(i)} \;\middle|\; \tau - \tau_{\text{last}} \leq i \leq \tau - 1 \right\}\right| > 1, \\ \text{wahr} & \text{sonst.} \end{cases}$$

Maximale Anzahl an Auswertungen der Fitnessfunktion Der Algorithmus terminiert, falls eine festgelegte Anzahl von Auswertungen der Fitnessfunktion überschritten wurde. Oft wird dieses Kriterium zusammen mit einer verzögerten Auswertung der Fitnessfunktion (siehe Abschnitt 4.4.6) umgesetzt. In diesem Fall ist das Kriterium nicht wie term_1 und term_2 deterministisch, da z. B. Mutationen probabilistisch angewendet werden und ein Individuum unverändert lassen können.

Im Gegensatz zu den anderen Terminierungsbedingungen hängt das Verhalten dieses Kriteriums von einem Implementierungsdetail, dem Einsatz der verzögerten Auswertung, ab. Wir führen es trotzdem auf, da es eine grobe Kontrolle über den Rechenaufwand[11] des Algorithmus ermöglicht und den Vergleich zwischen verschiedenen Parametersätzen (siehe Kapitel 5) des evolutionären Algorithmus erleichtert.

Sei $f_{\max} \in \mathbb{N}_0$. Wir setzen

$$\text{term}_3\big(\tau, f_{\text{eval}}^{(0)}, \ldots, f_{\text{eval}}^{(\tau-1)}\big) := \begin{cases} \text{falsch} & \text{falls } \sum_{i=0}^{\tau-1} f_{\text{eval}}^{(i)} \leq f_{\max}, \\ \text{wahr} & \text{sonst .} \end{cases}$$

Man beachte, dass der Algorithmus nicht unmittelbar nach der $f_{\max}$-ten Auswertung der Fitnessfunktion terminiert, da die Terminierungsbedingung nur zu Beginn einer Generation überprüft wird (siehe Algorithmus 4.1).

[11] Die Kontrolle über den Rechenaufwand basiert auf der Annahme, dass die Kosten zur Berechnung der Fitnessfunktion maßgeblich sind, was in unserem Fall nur bedingt zutrifft (siehe auch Abschnitt 4.4.6).

4.4.8 Selektion

Paarungsselektion und Umweltselektion verwenden die gleichen Selektionsmethoden, unterscheiden sich jedoch hinsichtlich deren Verwendung. Bei der Paarungsselektion kann ein Individuum im Verlauf der Selektion mehrfach ausgewählt werden (Selektion mit Duplikaten), während dies bei der Umweltselektion nur einmal möglich ist (duplikatfreie Selektion). Weitere Unterschiede liegen in der Art der Ausführung der Selektionsmethoden, den zur Selektion zur Verfügung stehenden Individuen sowie in der Anzahl auszuwählender Individuen. Tabelle 4.3 gibt einen Überblick.

Eigenschaft	Paarungsselektion	Umweltselektion
Duplikate erlaubt	ja	nein
Ausführung der Selektionsmethoden	blockweise	schrittweise
Auswahl aus	Eltern	(μ, λ) Selektion: Kinder $(\mu + \lambda)$ Selektion: Eltern und Kinder
Anzahl Selektionen	Summe über Anforderungen der Operatoren	μ

Tabelle 4.3: Unterschiede zwischen Paarungsselektion und Umweltselektion

4.4.8.1 Paarungsselektion

Ziel der Paarungsselektion ist die Auswahl von Individuen für die Rekombination; damit bestimmt sie durch die Auswahl der Eltern die Kinder einer Generation wesentlich mit.

In Anlehnung an die Verhältnisse in der Natur erlauben wir bei der Paarungsselektion die mehrfache Auswahl eines Individuums (dieses wird dementsprechend mehrfach zur Rekombination verwendet). Bei der Paarungsselektion wird aus den μ Elternindividuen einer Generation gewählt.

Die Anzahl der auszuwählenden Individuen ist abhängig von der Anzahl λ der zu erzeugenden Kinder sowie den Stelligkeiten der beteiligten Rekombinationsoperatoren. Wir nehmen vereinfachend an, dass jeder Rekombinationsoperator genau ein Kind erzeugt und die Anzahl der von jedem Rekombinationsoperator benötigten Individuen feststeht.[12] Für jedes der λ zu erzeugenden Kinder wird ein Rekombinationsoperator aufgerufen; die Anzahl der durch die Paarungsselektion auszuwählenden Individuen ist die Summe über die Anzahl der benötigten Individuen jedes eingesetzten Operators. Der Aufruf der Selektionsmethoden erfolgt blockweise, was für die Implementierung von Methoden wie der deterministischen Selektion vorteilhaft sein kann. Algorithmus 4.5 fasst das Vorgehen zusammen.

[12] Diese Annahmen schließen Rekombinationsoperatoren, die mehr als ein Individuum erzeugen und solche mit einer variablen Anzahl benötigter Individuen aus. Im Rahmen der Implementation kann man diesen Umstand wie folgt umgehen:

Rekombinationsoperatoren, die $k > 1$ Individuen erzeugen, können diese intern zwischenspeichern und bei ihren $k - 1$ nächsten Aufrufen zurückgeben, ohne tatsächlich neue Rekombinationen auszuführen. Ist $l \in \mathbb{N}$ die Anzahl der Aufrufe des Rekombinationsoperators, so wird nur ein $\lceil l/k \rceil / l$ Bruchteil der für diesen Operator ausgewählten Individuen tatsächlich verwendet. Um Verzerrungen zu vermeiden, sollten die von den Selektionsmethoden ausgewählten Individuen ebenfalls intern zwischengespeichert und in ihrer ursprünglichen Reihenfolge verwendet werden (so dass bei einer echten Rekombination nicht die aktuell übergebene Auswahl benutzt wird, sondern die erste noch nicht verwendete Auswahl). Diese Technik setzt voraus, dass den Rekombinationsoperatoren der Beginn einer neuen Generation signalisiert wird.

Rekombinationsoperatoren mit einer variablen Anzahl an benötigten Individuen können die maximal benötigte Anzahl an Individuen anfordern und ungenutzte Individuen intern zwischenspeichern, um diese dann, ähnlich wie im vorherigen Absatz, beim nächsten Aufruf einzusetzen.

Alternativ bestimmt man zuerst die Anzahl der von jedem Rekombinationsoperator zu erzeugenden Kinder, berechnet die Anzahl $\lceil l/k \rceil$ der notwendigen Aufrufe des Rekombinationsoperators und erhält so die genaue Anzahl benötigter Elternindividuen. Bei Rekombinationsoperatoren mit einer variablen Anzahl von Eltern muss diese vor der Paarungsselektion bestimmt werden.

Sei $l \in \mathbb{N}$ die Anzahl der Selektionsmethoden, $k \in \mathbb{N}$ die Anzahl der Rekombinationsoperatoren und s_i, $1 \leq i \leq k$ die Anzahl der vom i-ten Rekombinationsoperator benötigten Individuen.

(1) Führe Algorithmus 4.2 mit λ Ressourceneinheiten und den Rekombinationsoperatoren als Verbrauchern durch. Seien $a_1, \ldots, a_k$ die resultierenden Zuweisungen.

(2) Bestimme die Anzahl auszuwählender Individuen $s \leftarrow \sum_{i=1}^{k} a_i s_i$.

(3) Führe Algorithmus 4.2 mit s Ressourceneinheiten und den Selektionsmethoden als Verbrauchern durch. Wähle gemäß den resultierenden Zuordnungen blockweise Individuen aus.

Die so bestimmten Individuen werden als Eltern für die Rekombination verwendet.

ALGORITHMUS 4.5: Paarungsselektion für mehrere Selektionsmethoden

4.4.8.2 Umweltselektion

Ziel der Umweltselektion ist die Reduktion der Population auf μ Individuen. Je nachdem, welche Individuen zur Selektion zur Verfügung stehen, unterscheiden wir zwei Arten der Umweltselektion:

Bei der *Komma-Selektion* (auch (μ, λ)-Selektion) werden aus den λ Kindern der aktuellen Generation μ Individuen bestimmt, welche die Eltern der nächsten Generation bilden. Dazu muss $\lambda \geq \mu$ gelten. Bei der *Plus-Selektion* (auch $(\mu + \lambda)$-Selektion) werden aus allen $\mu + \lambda$ Individuen (Eltern und Kinder) der aktuellen Generation μ Individuen bestimmt, welche die Eltern der nächsten Generation bilden.

Wir verbieten bei der Umweltselektion die mehrfache Auswahl von Individuen. Es werden insgesamt μ Individuen ausgewählt. Kommen mehrere Selektionsmethoden zum Einsatz, wird die von jeder Methode zu bestimmende Anzahl an Individuen gemäß Algorithmus 4.2 bestimmt und die Selektionsmethoden schrittweise ausgeführt.

4.4.8.3 Selektionsmethoden

Ziel der Selektionsmethoden ist es, Individuen anhand ihrer Fitnesswerte auszuwählen; dabei werden Individuen mit höherer Fitness bevorzugt.

Um die Implementierung mancher Selektionsoperatoren zu vereinfachen, nehmen wir an, dass die zur Verfügung stehenden Individuen nach aufsteigender Fitness sortiert sind. Wir setzen dazu eine totale Ordnung auf den Fitnesswerten voraus. Im Folgenden geben wir verschiedene Selektionsmethoden, die jeweils ein Individuum I_{sel} aus einer gegebenen Folge $I_1, \ldots, I_k$ von Individuen mit $f(I_1) \leq \cdots \leq f(I_k)$ auswählen.

DETERMINISTISCHE SELEKTION DES BESTEN INDIVIDUUMS Das Individuum mit dem höchsten Fitnesswert wird ausgewählt, $I_{\text{sel}} = I_k$. Diese Methode übt den höchsten Selektionsdruck aus.

UNIFORME SELEKTION Das auszuwählende Individuum wird gleichverteilt aus allen Individuen gezogen: $\mathbb{P}[I_{\text{sel}} = I_p] = 1/k$. Dies ist die einzige Selektionsmethode, bei der die Fitness der Individuen keine Rolle spielt.

RANGBASIERTE SELEKTION Die Individuen werden zufällig gemäß einer Verteilung bestimmt; die Wahrscheinlichkeit eines Individuums, ausgewählt zu werden, hängt von seinem Rang, d. h. seiner Position in der nach Fitnesswerten sortierten Folge der Individuen, ab. Wir geben zwei rangbasierte Methoden zur Selektion: Lineare rangbasierte Selektion und exponentielle rangbasierte Selektion.

LINEARE RANGBASIERTE SELEKTION Die Wahrscheinlichkeit eines Individuums, ausgewählt zu werden, ist proportional zu seinem Rang:

$$\mathbb{P}[I_{\text{sel}} = I_p] := \frac{1}{k}\left(\eta^- + (\eta^+ - \eta^-)\frac{p-1}{k-1}\right), \quad 1 \leq p \leq k. \tag{4.2}$$

Da $\mathbb{P}[I_{\text{sel}} = I_1] = \eta^-/k$ und $\mathbb{P}[I_{\text{sel}} = I_k] = \eta^+/k$ bestimmen $\eta^-, \eta^+ \in [0, k] \subset \mathbb{R}$ die Wahrscheinlichkeiten für das schlechteste bzw. für das beste Individuum.

LEMMA 4.3 Die durch Gleichung 4.2 definierten Werte bilden für $\eta^+ = 2 - \eta^-$ eine Wahrscheinlichkeitsverteilung. □

BEWEIS Wegen $\eta^+, \eta^- \geq 0$ und $1 \leq p \leq k$ ist stets $\mathbb{P}[I_{\text{sel}} = I_p] \geq 0$ (der kleinste Wert, den der rechte Summand in der äußeren Klammer annehmen kann, ist $-\eta^-$, was stets durch den ersten Summanden kompensiert wird). Wir zeigen, dass sich die $\mathbb{P}[I_{\text{sel}} = I_p]$ genau für $\eta^+ = 2 - \eta^-$ zu 1 summieren:

$$\begin{aligned}
&\sum_{p=1}^{k} \mathbb{P}[I_{\text{sel}} = I_p] = 1 \\
\iff &\sum_{p=1}^{k} \frac{1}{k}\left(\eta^- + (\eta^+ - \eta^-)\frac{p-1}{k-1}\right) = 1 \\
\iff &\frac{1}{k}\left(k\eta^- + \frac{\eta^+ - \eta^-}{k-1}\sum_{p=1}^{k}(p-1)\right) = 1 \\
\iff &\frac{1}{k}\left(k\eta^- + \frac{\eta^+ - \eta^-}{k-1}\frac{k(k-1)}{2}\right) = 1 \\
\iff &\eta^- + \frac{\eta^+ - \eta^-}{2} = 1 \\
\iff &\eta^+ + \eta^- = 2 \\
\iff &\eta^+ = 2 - \eta^- \quad .
\end{aligned}$$

□

Setzt man $\eta^- := 2/(c+1)$ und $\eta^+ := 2c/(c+1)$, so ist die Bedingung aus Lemma 4.3 erfüllt und die Wahrscheinlichkeitsverteilung lässt sich über ihre Steigung $c \in \mathbb{R}_{\geq 0}$ kontrollieren. Abbildung 4.3 zeigt die entstehenden Verteilungen für einige Werte von c. Für $0 \leq c < 1$ haben schlechtere, für $c > 1$ bessere Individuen die größeren Wahrscheinlichkeiten. Für $c = 1$ ergibt sich die uniforme Verteilung als Spezialfall.

EXPONENTIELLE RANGBASIERTE SELEKTION Die Wahrscheinlichkeiten der Individuen werden exponentiell nach ihrem Rang gewichtet:

$$\mathbb{P}[I_{\text{sel}} = I_p] := \frac{c^{k-p}}{\sum_{i=1}^{k} c^{k-i}} = \frac{c-1}{c^k - 1}c^{k-p}, \quad 1 \leq p \leq k, \;\; 0 \leq c < 1, \tag{4.3}$$

wobei die Gleichheit aus $\sum_{i=1}^{k} c^{k-i} = \sum_{i=0}^{k-1} c^i = \frac{c^k-1}{c-1}$ folgt. Abbildung 4.4 zeigt die entstehenden Verteilungen für einige Werte von c. Für $c = 0$ ergibt sich die deterministische Selektion, für $c \to 1$ die uniforme Selektion. Man beachte, dass für den ersten der beiden Ausdrücke in Gleichung 4.3 der Wert $c = 1$ zulässig ist.

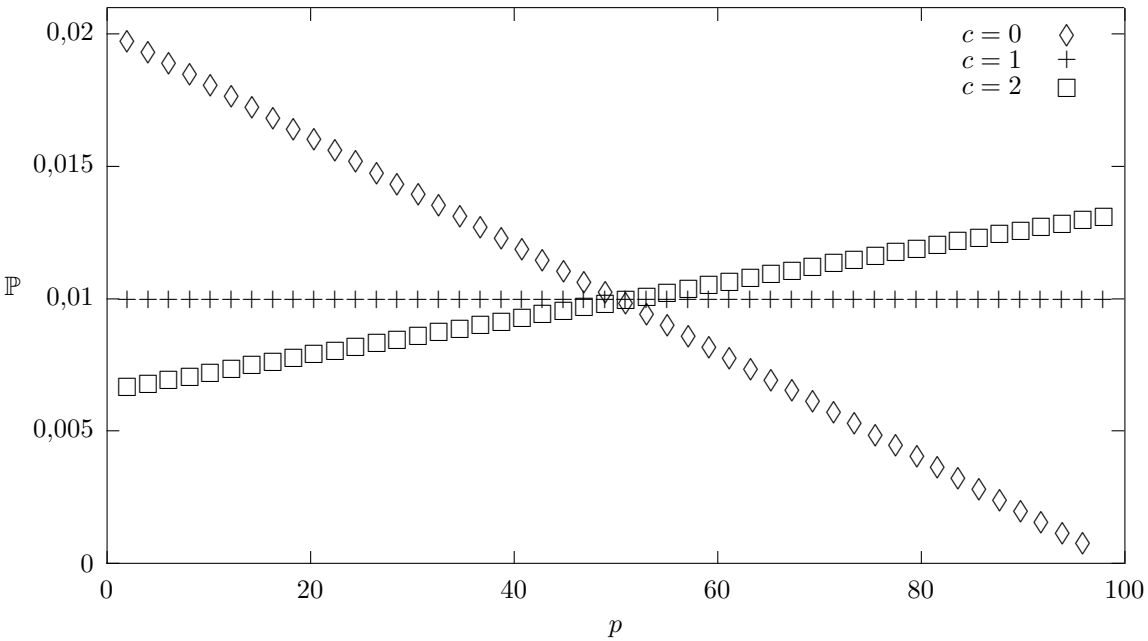


Abbildung 4.3: Lineare rangbasierte Selektion für $c \in \{0, 1, 2\}$ und $k = 100$

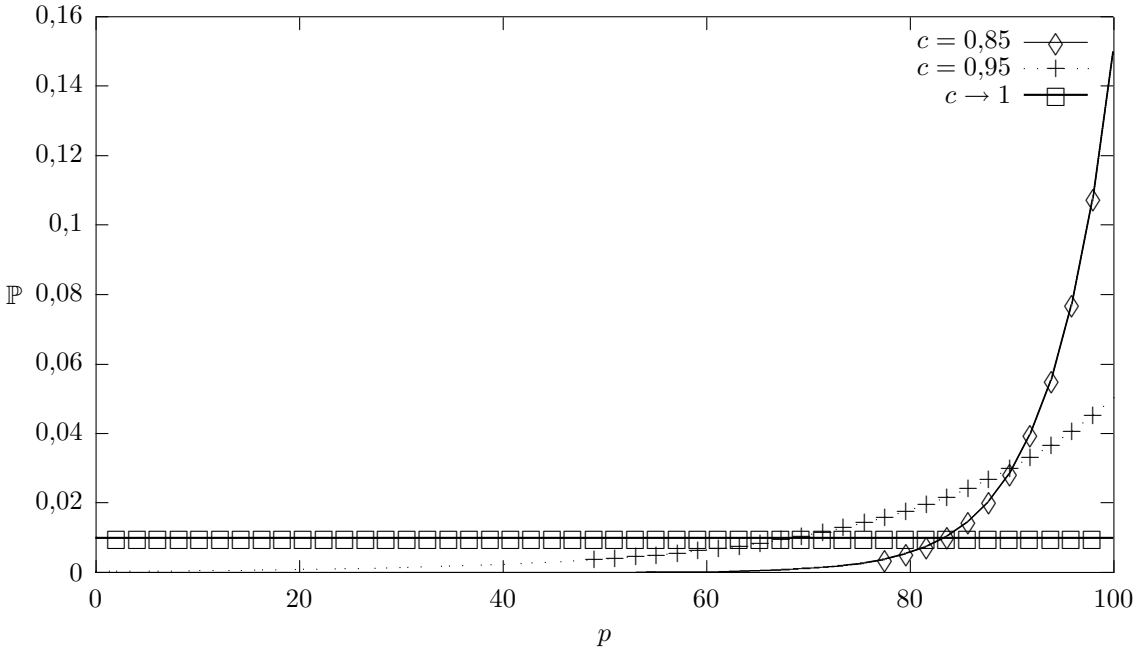


Abbildung 4.4: Exponentielle rangbasierte Selektion für $c \in \{0{,}85, 0{,}95\}$, $c \to 1$ und $k = 100$

Stochastische Turnierselektion Es werden $q \in \mathbb{N}$ Individuen gleichverteilt mit Wiederholung aus allen Individuen gezogen und von diesen das beste ausgewählt. Die Reihenfolge, in der die Individuen gezogen werden, spielt keine Rolle. Die Anschauung dabei ist, dass die q Individuen ein Turnier austragen, bei dem in jeder Runde zwei Individuen gegeneinander antreten und das Individuum mit der niedrigeren Fitness ausscheidet. Der Gewinner des Turniers wird ausgewählt.

Da es nicht auf die absoluten Fitnesswerte der Individuen ankommt, sondern nur die Fitnesswerte von Individuen miteinander verglichen werden, ist die Methode letztlich rangbasiert. Es gibt mehrere Möglichkeiten, unterschiedliche Individuen mit gleicher Fitness zu behandeln. Algorithmus 4.6 zeigt zwei davon; andere sind die uniforme Auswahl oder die Voraussetzung eindeutiger Fitnesswerte. Abbildung 4.5 zeigt die Verteilung der stochastischen Turnierselektion für $q \in \{2, 5, 10\}$.

Bezeichne $I_1, \ldots, I_k$ die Individuen und sei $q \in \mathbb{N}$.

(1) Wähle I_{sel} gleichverteilt aus $\{I_1, \ldots, I_k\}$.

(2) Wiederhole $(q-1)$-mal:

  (a) Ziehe X gleichverteilt aus $\{I_1, \ldots, I_k\}$.

  (b) Falls $f(X) > f(I_{\text{sel}})$ dann setze $I_{\text{sel}} \leftarrow X$.

Sind die $I_1, \ldots, I_k$ sortiert, kann man sich auf die Ziehung von Indizes beschränken:

(1) Wähle sel gleichverteilt aus $\{1, \ldots, k\}$.

(2) Wiederhole $(q-1)$-mal:

  (a) Ziehe X gleichverteilt aus $\{1, \ldots, k\}$.

  (b) Falls $X >$ sel dann setze sel $\leftarrow X$.

Man beachte, dass Individuen mit gleicher Fitness unterschiedlich gehandhabt werden.

Algorithmus 4.6: Zwei Varianten der Turnierselektion

Andere Selektionsmethoden Bei der fitnessproportionalen Selektion werden die Individuen gemäß einer Verteilung bestimmt, deren Wahrscheinlichkeiten proportional zu den Fitnesswerten der Individuen sind. Wir verzichten aufgrund unerwünschter Effekte[13] auf diese Selektionsmethode.

4.4.9 Rekombination

Ein Rekombinationsoperator erstellt aus einem[14] oder mehreren Individuen, den Eltern, ein oder mehrere neue Individuen, die Kinder. Aufgrund der Annahmen bei der Paarungsselektion (siehe Abschnitt 4.4.8.1) beschränken wir uns auf Rekombinationsoperatoren mit einem Kind und einer festen Zahl an Eltern.

[13] Hat ein Individuum eine wesentlich höhere Fitness als alle anderen Individuen, so hat es auch eine wesentlich höhere Wahrscheinlichkeit, zur Rekombination herangezogen zu werden. Dies kann aufgrund des sehr hohen Selektionsdrucks die Diversität einer Population zu schnell verringern (übermäßiger Selektionsdruck). Haben alle Individuen annähernd gleiche Fitnesswerte, so haben sie auch annähernd gleiche Wahrscheinlichkeiten, wodurch die Individuen mit geringfügig höherer Fitness praktisch keinen Selektionsvorteil mehr haben (mangelnder Selektionsdruck). [48]

[14] Da Rekombinationsoperatoren — im Gegensatz zu Mutationsoperatoren — Informationen mehrerer Individuen zur Erstellung eines neuen Individuums verwenden, sind mindestens zwei Eltern beteiligt; einzige Ausnahme ist der Identitätsoperator. Operatoren, die ein einzelnes Individuum auf Basis der restlichen Population nur verändern, aber nicht neu erstellen, sind keine Rekombinationsoperatoren im eigentlichen Sinne (siehe auch Abschnitt 4.4.10.4).

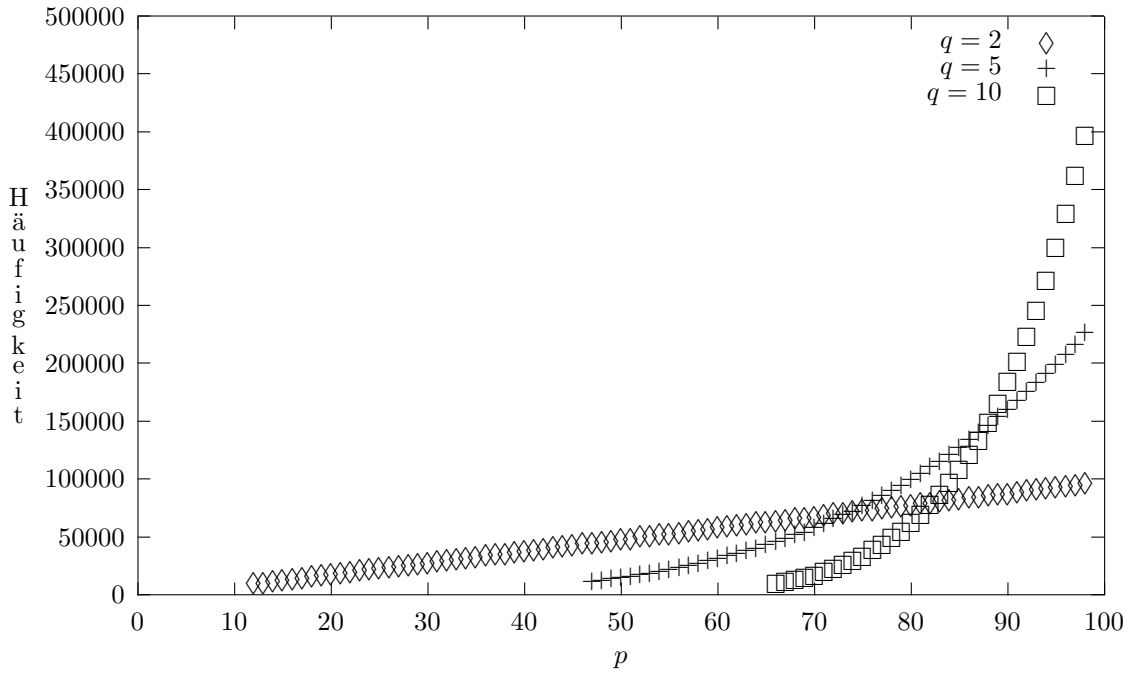


Abbildung 4.5: Stochastische Turnierselektion für $k = 100$, $q \in \{2, 5, 10\}$ und 5 000 000 durchgeführte Selektionen

4.4.9.1 Identität

Dieser Rekombinationsoperator benötigt ein Elternindividuum, welches unverändert als Kind übernommen wird. Die Identität als Rekombinationsoperator erlaubt die Beschränkung auf Mutationsoperatoren, ohne Algorithmus 4.1 strukturell zu verändern.

4.4.9.2 Kantenrekombination

Dieser für das WCP-A entworfene Rekombinationsoperator basiert auf einem Kantenrekombinationsoperator für das TSP [51]. Der Rekombinationsoperator für das WCP-A erlaubt die Verwendung einer variablen Anzahl von Elternindividuen, wobei ein Individuum mehrmals als Elternteil auftreten darf. Er ist darauf ausgelegt, die Nachbarschaftsverhältnisse der Knoten in den Touren der Eltern zu erhalten. Es handelt sich um ein zweistufiges Verfahren:

Zuerst werden die Knoten probabilistisch auf die Partitionen verteilt; die Wahrscheinlichkeit eines Knotens, der i-ten Partition zugeordnet zu werden, ist dabei proportional zur Anzahl des Auftretens dieser Zuordnung in den Elternindividuen.

Anschließend werden für jede Partition die ihr zugewiesenen Knoten angeordnet. Man erstellt dazu eine Tabelle, in die man für jeden Knoten v seine Nachbarn in den Touren der Eltern einträgt; dabei berücksichtigt man nur diejenigen Nachbarn, die der gleichen Partition wie v zugeordnet wurden. Tritt ein solcher Knoten in mehreren Partitionen der Eltern als Nachbar von v auf, erscheint er auch mehrfach im Tabelleneintrag für v.

Ausgehend vom Startknoten bestimmt man den nächsten Knoten wie folgt: Unter allen Knoten im Tabelleneintrag des aktuell letzten Knotens wählt man den am häufigsten vorkommenden Knoten aus; bei gleich häufigen Knoten wählt man den Knoten mit dem seinerseits kleinsten Tabelleneintrag aus. Gibt es auch hier mehrere Kandidaten, lässt man den Zufall entscheiden.

Die Idee ist, in der ersten Stufe des Verfahrens die Zuordnungen der Knoten zu den Partitionen

und in der zweiten Stufe die Nachbarschaftsbeziehungen der Knoten untereinander zu erhalten. Algorithmus 4.7 beschreibt den Kantenrekombinationsoperator für das WCP-A im Detail. Für $m = 1$ ergibt sich als Spezialfall ein Rekombinationsoperator für das TSP. Der Algorithmus verwendet als primäres Kriterium (Schritt 3bii) für die Knotenwahl die Anzahl der Vorkommen eines Knotens in der Multimenge. Dadurch werden gemeinsame Teilsequenzen der Eltern bevorzugt. Das sekundäre Auswahlkriterium (Schritt 3biii) ist die Größe der Tabelleneinträge der Kandidaten. Durch die bevorzugte Wahl von Knoten mit wenigen verfügbaren Nachbarn werden Engstellen vermieden und die Fortsetzbarkeit des Verfahrens ohne Rückgriff auf Schritt 3bi wird wahrscheinlicher. Führt keines der genannten Kriterien zu einem eindeutigen Ergebnis, wird in Schritt 3biv einer der in Frage kommenden Knoten zufällig ausgewählt.

Beispiel 4.4 (Kantenrekombination) Sei $m = 2$, $n = 7$ und $p = 3$. Die Elternindividuen seien durch $I_1 = \big((5,1,7,3),(2,4,6)\big)$, $I_2 = \big((5),(3,7,1,2,4,6)\big)$ und $I_3 = \big(6,3,7,1,5,4,2),()\big)$ gegeben.

Zuerst verteilen wir in Schritt 1 die Knoten auf die Partitionen. Es ist $\mathbb{P}[X_1 = 1] = \frac{1}{3}(1 + 0 + 1)$, $\mathbb{P}[X_1 = 2] = \frac{1}{3}(0 + 1 + 0)$ usw. Tabelle 4.4 zeigt die sich ergebenden Wahrscheinlichkeiten. Seien $(X_j)_{1 \le j \le n} = (1,2,1,2,1,1,1)$ die Ergebnisse der Zufallsexperimente. Wir erhalten die Mengen $L'_1 = \{1,3,5,6,7\}$ und $L'_2 = \{2,4\}$.

In Schritt 2 erstellen wir die leeren Multimengen $Q_1, \ldots, Q_7$ sowie R_1, R_2 und führen die in Tabelle 4.5 aufgezählten Vereinigungen durch. Es ergeben sich die Mengen

$$\begin{array}{ll} Q_1 = \{5,5,7,7,7\}, & Q_5 = \{1,1\}, \\ Q_2 = \{4,4,4\}, & Q_6 = \{3\}, \\ Q_3 = \{6,7,7,7\}, & Q_7 = \{1,1,1,3,3,3\}, \\ Q_4 = \{2,2,2\}, & \\ R_1 = \{3,5,5,6\}, & R_2 = \{2\}\ . \end{array}$$

Wir beginnen Schritt 3 mit der Partition $i = 1$. Da $|L'_1| = 5 > 0$ wählen wir zuerst in Schritt 3a den ersten Knoten von L_1. Wegen $R_1 \neq \emptyset$ bestimmen wir, analog zu Schritt 3bii, $Q = \arg\max\{\#(R_1, l) \mid l \in R_1\} = \{5\}$ und somit $l = 5$. Wir setzen $L_{1,1} \leftarrow 5$, entfernen 5 aus Q_1 sowie L'_1 und fahren mit Schritt 3b fort. Da $|L'_1| = 4 > 0$ setzen wir $j \leftarrow L_{1,|L_1|} = 5$. In Schritt 3bi ist $Q_5 \neq \emptyset$ und wir fahren mit Schritt 3bii fort. Dort ist $Q = \{1\}$, wir setzen $l \leftarrow 1$, in Schritt 3bv dann $L_{1,2} \leftarrow 1$, $L'_1 \leftarrow \{3{,}6{,}7\}$ und entfernen 1 aus Q_5 und Q_7. In Schritt 3b ist $|L'_1| = 3 > 0$, $j \leftarrow 1$, $Q_1 \neq \emptyset$, $Q = \{7\}$, wir setzen $L_{1,3} \leftarrow 7$ und entfernen 7 aus L'_1, Q_1 und Q_3. Mit $|L'_1| = 2 > 0$, $j \leftarrow 7$, $Q_7 \neq \emptyset$, $Q = \{3\}$ setzen wir $L_{1,4} \leftarrow 3$, $L'_1 \leftarrow \{6\}$ und entfernen 3 aus L'_1, Q_6 und Q_7. Im letzten Durchlauf ist $|L'_1| = 1 > 0$, $j \leftarrow 3$, $Q_3 \neq \emptyset$, $Q = \{6\}$ und $L_{1,6} \leftarrow 6$. Wir entfernen 6 aus L'_1 und Q_3.

Da $|L'_1| = 0$ und $|L'_2| = 2 > 0$ beginnen wir mit Schritt 3a für Partition $i = 2$. Wir erhalten analog zu Schritt 3bii als ersten Knoten $l \leftarrow 2$, setzen $L_{2,1} \leftarrow 2$ und entfernen 2 aus L'_2 und Q_4. Da $|L'_2| = 1 > 0$ setzen wir $j \leftarrow 2$ und erhalten in Schritt 3bii die Menge $Q = \{4\}$, also $L_{2,2} \leftarrow 4$. Wir entfernen 4 aus L'_2 sowie Q_2 und der Algorithmus terminiert.

Insgesamt erhalten wir als Ergebnis $L = \big((5,1,7,3,6),(2,4)\big)$. Man beachte den Erhalt der allen Eltern gemeinsamen Teilsequenz $(1,7,3)$. □

	Knoten j						
Partition i	1	2	3	4	5	6	7
1	$\frac{2}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{2}{3}$
2	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	0	$\frac{2}{3}$	$\frac{1}{3}$

Tabelle 4.4: Wahrscheinlichkeiten $\mathbb{P}[X_j = i]$ der Knotenzuordnungen in Beispiel 4.4

Seien $1, \ldots, n$, $s_1, \ldots, s_m$ die Knoten des Graphen, $I_1, \ldots, I_p$, $p \in \mathbb{N}$ die Elternindividuen und L das zu erstellende Kindindividuum.

(1) *Verteilung der Knoten auf die Partitionen:*
Führe Zufallsexperimente $X_1, \ldots, X_n \in \{1, \ldots, m\}$ mit

$$\mathbb{P}[X_j = i] := \frac{1}{p} \sum_{k=1}^{p} 1_{i,j,k}, \qquad 1_{i,j,k} := \begin{cases} 1 & \text{falls } \exists l \in \mathbb{N} : \big(I_k\big)_{i,l} = j, \\ 0 & \text{sonst} \end{cases}$$

durch und ordne Knoten j der Menge L'_{X_j} zu.

(2) *Erstellen der Kantentabelle:*
Es bezeichne $L' : \{1, \ldots, n\} \to \{L'_l \mid 1 \le l \le m\}$ mit $L'(j) = L'_i$ falls $j \in L'_i$ die Funktion, die jeden Knoten seiner Menge zuordnet.

(a) Erstelle für jeden Knoten j eine leere Multimenge Q_j.
$\forall k \in \{1, \ldots, p\} \forall i \in \{1, \ldots, m\} \forall j \in \{1, \ldots, |(I_k)_i| - 1\}$ setze
$Q_{(I_k)_{i,j}} \leftarrow Q_{(I_k)_{i,j}} \cup \Big(\big\{(I_k)_{i,j+1}\big\} \cap L'\big((I_k)_{i,j}\big) \Big)$ und
$Q_{(I_k)_{i,j+1}} \leftarrow Q_{(I_k)_{i,j+1}} \cup \Big(\big\{(I_k)_{i,j}\big\} \cap L'\big((I_k)_{i,j+1}\big) \Big)$.

(b) Erstelle für jeden Startknoten s_i eine leere Multimenge R_i.
$\forall k \in \{1, \ldots, p\} \forall i \in \{1, \ldots, m\}$ falls $|(I_k)_i| > 0$ setze
$R_i \leftarrow R_i \cup \Big(\big\{(I_k)_{i,1}, (I_k)_{i,|(I_k)_i|}\big\} \cap L'_i \Big)$

(3) *Konstruktion der Touren:*
Für jede Partition $i \in \{1, \ldots, m\}$ von L mit $|L'_i| > 0$:

(a) Wähle den ersten Knoten von L_i gemäß den Schritten 3bi bis 3bv, wobei außer in Schritt 3bv die Q_j durch R_i zu ersetzen sind.

(b) Solange $|L'_i| > 0$ wiederhole:
Sei $j \leftarrow L_{i,|L_i|}$ der zum jetzigen Zeitpunkt letzte Knoten von L_i.
Es bezeichne $\#(Q_j, l)$ die Anzahl der Vorkommen von l in der Multimenge Q_j.

(i) Falls $Q_j = \emptyset$ dann wähle gleichverteilt einen Knoten $l \in L_i$ und gehe zu Schritt 3bv.

(ii) Setze die Menge $Q \leftarrow \arg\max \{\#(Q_j, l) \mid l \in Q_j\}$.
Falls $|Q| = 1$ setze $l \leftarrow l \in Q$ und gehe zu Schritt 3bv.

(iii) Setze $Q \leftarrow Q \cap \arg\min \{|Q_l| \mid l \in Q\}$.
Falls $|Q| = 1$ setze $l \leftarrow l \in Q$ und gehe zu Schritt 3bv.

(iv) Wähle l gleichverteilt aus Q.

(v) Setze $L_{i,|L_i|+1} \leftarrow l$, $L'_i \leftarrow L'_i \setminus \{l\}$, $j \leftarrow l$ und entferne sämtliche Vorkommen von l in allen Q_j, $1 \le j \le n$.

Wir gehen davon aus, dass arg min und arg max die Menge aller minimalen bzw. maximalen Argumente liefern. Schritt 1 kann implementiert werden, indem für jeden Knoten gleichverteilt eines der Elternindividuen gewählt wird und der Knoten der gleichen Partition wie im Elternindividuum zugeordnet wird.

Algorithmus 4.7: Kantenrekombination für das WCP-A

k	i	j	Vereinigungen	
1	1	1	$Q_5 \leftarrow Q_5 \cup (\{1\} \cap L'_1) = Q_5 \cup \{1\}$	$Q_1 \leftarrow Q_1 \cup (\{5\} \cap L'_1) = Q_1 \cup \{5\}$
1	1	2	$Q_1 \leftarrow Q_1 \cup (\{7\} \cap L'_1) = Q_1 \cup \{7\}$	$Q_7 \leftarrow Q_7 \cup (\{1\} \cap L'_1) = Q_7 \cup \{1\}$
1	1	3	$Q_7 \leftarrow Q_7 \cup (\{3\} \cap L'_1) = Q_7 \cup \{3\}$	$Q_3 \leftarrow Q_3 \cup (\{7\} \cap L'_1) = Q_3 \cup \{7\}$
1	2	1	$Q_2 \leftarrow Q_2 \cup (\{4\} \cap L'_2) = Q_2 \cup \{4\}$	$Q_4 \leftarrow Q_4 \cup (\{2\} \cap L'_2) = Q_4 \cup \{2\}$
1	2	2	$Q_4 \leftarrow Q_4 \cup (\{6\} \cap L'_2) = Q_4$	$Q_6 \leftarrow Q_6 \cup (\{4\} \cap L'_1) = Q_6$
2	2	1	$Q_3 \leftarrow Q_3 \cup (\{7\} \cap L'_1) = Q_3 \cup \{7\}$	$Q_7 \leftarrow Q_7 \cup (\{3\} \cap L'_1) = Q_7 \cup \{3\}$
2	2	2	$Q_7 \leftarrow Q_7 \cup (\{1\} \cap L'_1) = Q_7 \cup \{1\}$	$Q_1 \leftarrow Q_1 \cup (\{7\} \cap L'_1) = Q_1 \cup \{7\}$
2	2	3	$Q_1 \leftarrow Q_1 \cup (\{2\} \cap L'_1) = Q_1$	$Q_2 \leftarrow Q_2 \cup (\{1\} \cap L'_2) = Q_2$
2	2	4	$Q_2 \leftarrow Q_2 \cup (\{4\} \cap L'_2) = Q_2 \cup \{4\}$	$Q_4 \leftarrow Q_4 \cup (\{2\} \cap L'_2) = Q_4 \cup \{2\}$
2	2	5	$Q_4 \leftarrow Q_4 \cup (\{6\} \cap L'_2) = Q_4$	$Q_6 \leftarrow Q_6 \cup (\{4\} \cap L'_1) = Q_6$
3	1	1	$Q_6 \leftarrow Q_6 \cup (\{3\} \cap L'_1) = Q_6 \cup \{3\}$	$Q_3 \leftarrow Q_3 \cup (\{6\} \cap L'_1) = Q_3 \cup \{6\}$
3	1	2	$Q_3 \leftarrow Q_3 \cup (\{7\} \cap L'_1) = Q_3 \cup \{7\}$	$Q_7 \leftarrow Q_7 \cup (\{3\} \cap L'_1) = Q_7 \cup \{3\}$
3	1	3	$Q_7 \leftarrow Q_7 \cup (\{1\} \cap L'_1) = Q_7 \cup \{1\}$	$Q_1 \leftarrow Q_1 \cup (\{7\} \cap L'_1) = Q_1 \cup \{7\}$
3	1	4	$Q_1 \leftarrow Q_1 \cup (\{5\} \cap L'_1) = Q_1 \cup \{5\}$	$Q_5 \leftarrow Q_5 \cup (\{1\} \cap L'_1) = Q_5 \cup \{1\}$
3	1	5	$Q_5 \leftarrow Q_5 \cup (\{4\} \cap L'_1) = Q_5$	$Q_4 \leftarrow Q_4 \cup (\{5\} \cap L'_2) = Q_4$
3	1	6	$Q_4 \leftarrow Q_4 \cup (\{2\} \cap L'_2) = Q_4 \cup \{2\}$	$Q_2 \leftarrow Q_2 \cup (\{4\} \cap L'_2) = Q_2 \cup \{4\}$

(a) Multimengen Q_j der Knoten

k	i	Vereinigung
1	1	$R_1 \leftarrow R_1 \cup (\{5,3\} \cap L'_1) = R_1 \cup \{5,3\}$
1	2	$R_2 \leftarrow R_2 \cup (\{2,6\} \cap L'_2) = R_2 \cup \{2\}$
2	1	$R_1 \leftarrow R_1 \cup (\{5,5\} \cap L'_1) = R_1 \cup \{5\}$
2	2	$R_2 \leftarrow R_2 \cup (\{3,6\} \cap L'_2) = R_2$
3	1	$R_1 \leftarrow R_1 \cup (\{6,2\} \cap L'_1) = R_1 \cup \{6\}$
3	2	$\lvert(I_3)_2\rvert = 0$

(b) Multimengen R_i der Startknoten

Tabelle 4.5: Erstellung der Kantentabelle in Beispiel 4.4

4.4.10 Mutation

Ein Mutationsoperator verändert ein einzelnes Individuum. Betrifft die Veränderung nur einen kleinen Teil des Individuums und erfolgt sie ohne die Verwendung weiterer Informationen, sprechen wir von einem lokalen Zug (engl. local move). Von den im Folgenden vorgestellten Mutationsoperatoren sind das simulierte Abkühlen (siehe Abschnitt 4.4.10.5) sowie die populationsgesteuerte Inversion und Transposition (siehe Abschnitt 4.4.10.4) keine lokalen Züge in diesem Sinne.

Ein Mutationsoperator definiert einen Nachbarschaftsgraphen, bei dem die Knoten den möglichen Individuen entsprechen; zwei Individuen sind genau dann durch eine Kante miteinander verbunden, wenn es eine Mutation gibt, die sie ineinander überführt. Der Graph ist ungerichtet, falls es für jede Mutation π eine Mutation π' mit $\pi'(\pi(L)) = L$ gibt.

4.4.10.1 Identität

Das Individuum bleibt unverändert.

Die Identität als Mutationsoperator erlaubt den Verzicht auf eine Mutationswahrscheinlichkeit.[15] Verwendet man Algorithmus 4.2 zur Kombination mehrerer Mutationsoperatoren, entspricht die Erhöhung bzw. Verringerung der relativen Anforderung des Identitätsoperators der Erhöhung bzw. Verringerung der Mutationswahrscheinlichkeit. Da der Algorithmus weitgehend deterministisch arbeitet, ist die Entsprechung nur bei unendlich vielen Mutationskandidaten exakt und wird mit abnehmender Zahl der Mutationskandidaten immer schlechter.

4.4.10.2 Inversion

Unter einer Inversion verstehen wir die Umkehrung der Reihenfolge der Elemente innerhalb eines zusammenhängenden Abschnitts einer Sequenz von Elementen, z. B. den Knoten einer angeordneten Partition eines Individuums. Ähnlich wie im Beweis zur Suchraumgröße des WCP-A (siehe Seite 39) stellt sich die Frage, ob man diejenigen Inversionen einer Partition, die in gleichen Nachbarschaftsverhältnissen der Elemente resultieren, als äquivalent ansieht.

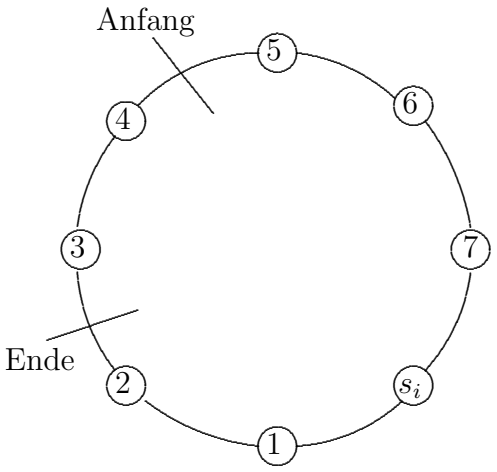
Zur Verdeutlichung betrachten wir eine angeordnete Partition L_i in der Kodierung eines Individuums: Fasst man L_i, zusammen mit dem Startknoten, als Tour auf, teilt diese Tour durch die Wahl von zwei Kanten als Trennstellen in zwei Hälften und kehrt die Reihenfolge der Knoten in einer der beiden Hälften um, erhält man eine Inversion dieser Tour. Für die Nachbarschaftsverhältnisse der Knoten spielt es keine Rolle, in welcher Hälfte der Tour die Knoten umgekehrt werden; Abbildung 4.6 verdeutlicht dies an einem Beispiel. Ein solches Paar von Inversionen unterscheidet sich nur in der Richtung der resultierenden Tour voneinander. Während diese beim TSP und beim WCP-A für die Güte einer Lösung ohne Bedeutung ist, gilt dies bei schwierigen Varianten des Schweißzellenproblems nicht mehr. Wir entscheiden uns mit der gleichen Begründung wie im Beweis zur Suchraumgröße des WCP-A für die Unterscheidung solcher Inversionen.

Lemma 4.4 Es gibt genau zwei Touren mit gleichem Startknoten und gleichen Nachbarschaftsverhältnissen der Knoten. □

Beweis Ist $\pi = (s_i, \pi_1, \pi_2, \ldots, \pi_l)$ eine Tour, dann auch $(s_i, \pi_l, \pi_{l-1}, \ldots, \pi_1)$ und die Nachbarschaftsverhältnisse sind gleich (jeder Knoten hat die gleichen Nachbarn). Jede weitere Tour, die den Bedingungen genügen soll, muss die gleiche Länge haben, da sich sonst die Nachbarschaftsverhältnisse an einem fehlenden oder zusätzlichen Knoten unterscheiden.

Angenommen $\pi' = (s_i, \pi'_1, \pi'_2, \ldots, \pi'_l)$ wäre eine solche weitere Tour. Da s_i in π die Nachbarn π_1 und π_l hat, muss s_i diese Nachbarn auch in π' haben, d. h. $\pi'_1 \in \{\pi_1, \pi_l\}$. Ist $\pi'_1 = \pi_1$, dann

[15] Der Begriff Mutationswahrscheinlichkeit wird unterschiedlich verwendet. Wir verstehen darunter an dieser Stelle eine Wahrscheinlichkeit für die Anwendung eines beliebigen Mutationsoperators auf ein durch einen Rekombinationsoperator erzeugtes Individuum.



Invertiert man die kleinere Hälfte der Tour, erhält man als Inversion

$$(s_i, 1, 2, 3, 4, 5, 6, 7) \mapsto (s_i, 1, 2, 4, 3, 5, 6, 7).$$

Bei Inversion der größeren Hälfte der Tour ergibt sich

$$(s_i, 1, 2, 3, 4, 5, 6, 7) \mapsto (s_i, 7, 6, 5, 3, 4, 2, 1).$$

Die Nachbarschaftsverhältnisse der Knoten sind in beiden resultierenden Touren gleich.

Abbildung 4.6: Äquivalenz von Inversionen

muss $\pi'_2 = \pi_2$ sein, da π_1 in π die Nachbarn s_i, π_2 hat und π'_1 bereits s_i als Nachbarn hat. Ist $\pi'_2 = \pi_2$, dann muss $\pi'_3 = \pi_3$ sein usw. bis $\pi'_l = \pi_l$. Damit ist $\pi' = \pi$, Widerspruch. Für den Fall $\pi'_1 = \pi_l$ argumentiert man analog. □

Basierend auf Lemma 4.4 entwerfen wir einen Algorithmus zur uniformen Inversion einer Partition eines Individuums. Dazu nummerieren wir die Trennstellen wie in Abbildung 4.7 gezeigt. Wir ziehen gleichverteilt zwei Trennstellen, von denen wir die erste als Anfang und die zweite als Ende der Inversion markieren. Es können zwei Fälle eintreten: Hat die Anfangsmarkierung einen kleineren Index als die Endmarkierung, wird der entsprechende Abschnitt direkt invertiert. Im anderen Fall geht die Inversion über den Startknoten hinweg. Wir führen dies auf zwei Inversionen des ersten Typs zurück, indem wir die beiden Trennstellen vertauschen, wie im ersten Fall invertieren und dann die gesamte Partition mit Ausnahme des Startknotens in der Reihenfolge umkehren. Sind Anfangs- und Endmarkierung identisch, geschieht nichts (leere Inversion). Algorithmus 4.8 beschreibt das Verfahren im Detail.

	1		2		3		4		$\lvert L_i\rvert-1$		$\lvert L_i\rvert$		$\lvert L_i\rvert+1$
s_i		1		2		3		·····		$\lvert L_i\rvert-1$		$\lvert L_i\rvert$	

Abbildung 4.7: Trennstellen für die Inversion

Der Mutationsoperator Inversion wählt zuerst gleichverteilt eine Partition des Individuums aus und invertiert anschließend gemäß Algorithmus 4.8.

Beispiel 4.5 (Mutationsoperator Inversion) Das Individuum $L = \big((5, 1, 7, 3, 6), (2, 4)\big)$ sei Argument des Mutationsoperators. Zur Bestimmung der Partition ziehen wir zuerst gleichverteilt $i \in \{1, 2\}$ mit Ergebnis $i = 1$. Anschließend ziehen wir in Schritt 1 des Algorithmus gleichverteilt $p, q \in \{1, \ldots, 6\}$ mit Ergebnis $p = 4$, $q = 1$. Da $p = 4 > 1 = q$ gehen wir zu Schritt 2b. Dort vertauschen wir p und q, iterieren j in Schritt 2a von 1 bis $\lfloor (4 - 1)/2 \rfloor = 1$ und vertauschen somit $L_{1,1+1-1}$ und $L_{1,4-1}$; als Zwischenschritt ergibt sich die Partition $(7, 1, 5, 3, 6)$. In Schritt 2c führen wir Schritt 2a für $p = 1$ und $q = 6$ aus. Wir iterieren j von 1 bis $\lfloor (6 - 1)/2 \rfloor = 2$ und vertauschen $L_{1,1+1-1}$ und $L_{1,6-1}$ sowie L_{1+2-1} und L_{6-2}. Als Ergebnis erhalten wir das Individuum $L' = \big((6, 3, 5, 1, 7), (2, 4)\big)$. □

Sei L_i eine Partition eines Individuums.

(1) Wähle p, q gleichverteilt aus $\{1, \ldots, |L_i| + 1\}$.

(2) Falls $p \leq q$ dann

(a) Iteriere j von 1 bis $\lfloor (q-p)/2 \rfloor$ und vertausche dabei $L_{i,p+j-1} \leftrightarrow L_{i,q-j}$.

ansonsten

(b) Vertausche $p \leftrightarrow q$ und führe Schritt 2a aus.

(c) Setze $p \leftarrow 1$, $q \leftarrow |L_i| + 1$ und führe Schritt 2a aus.

In Schritt 2a werden die Positionen $p, p+1, \ldots, q-1$ in ihrer Reihenfolge umgekehrt.

ALGORITHMUS 4.8: Inversion einer angeordneten Partition eines Individuums

Im Gegensatz zum TSP lässt sich beim WCP-A nicht der gesamte Suchraum durch Inversionen erreichen, da diese auf Veränderungen innerhalb einer Partition beschränkt sind. Wir stellen daher im nächsten Abschnitt einen Mutationsoperator vor, der die Verschiebung von Knoten zu anderen Partitionen ermöglicht.

4.4.10.3 Transposition

Unter einer Transposition verstehen wir die Verschiebung der Elemente eines zusammenhängenden Abschnitts innerhalb einer Sequenz von Elementen, z. B. den Knoten einer angeordneten Partition eines Individuums. Die Verschiebung kann sowohl innerhalb einer Sequenz als auch von einer Sequenz zu einer anderen Sequenz erfolgen. Wir beschränken uns beim Entwurf des Mutationsoperators auf den Fall einelementiger Abschnitte, d. h. auf die Verschiebung einzelner Knoten. Wir unterscheiden zwei Arten der Transposition:

TRANSPOSITION ZWISCHEN PARTITIONEN Wir bestimmen zufällig eine Partition und einen Knoten in dieser Partition, entfernen diesen, bestimmen zufällig eine weitere Partition sowie eine Trennstelle und fügen den Knoten an dieser Trennstelle ein. Algorithmus 4.9 beschreibt das Vorgehen im Detail.

TRANSPOSITION INNERHALB EINER PARTITION Die Transposition eines Knotens innerhalb einer Partition ergibt sich als Spezialfall für $i = k$ in Algorithmus 4.9. Beispiel 4.6 beinhaltet auch diesen Fall.

BEISPIEL 4.6 (TRANSPOSITION) Sei $L = \big((1,5,6,4,12,11,9),(),(7,2,3,10,8)\big)$ ein Individuum. Wir ziehen in Schritt 1 die Zahlen $i = 1$ und $p = 7$. In Schritt 2 setzen wir $v \leftarrow L_{1,7} = 9$ und $L_1 \leftarrow (L_{1,1}, \ldots, L_{1,6}, L_{1,8}, \ldots, L_{1,7}) = (1,5,6,4,12,11)$. Ziehen wir in Schritt 3 die Partition $k = 2$ und die Trennstelle $q = 1 \in \{1, \ldots, |L_2| + 1\} = \{1\}$, setzen wir im nächsten Schritt $L_2 = (L_{2,1}, L_{2,2}, \ldots, L_{2,0}, v, L_{2,1}, L_{2,2}, \ldots, L_{2,0}) = (9)$ und erhalten als Ergebnis $L' = \big((1,5,6,4,12,11),(9),(7,2,3,10,8)\big)$.

Hätten wir in Schritt 3 die Partition $k = 1$ und die Trennstelle $q = 1$ gezogen, würden wir $L_1 \leftarrow (L_{1,1}, L_{1,2}, \ldots, L_{1,0}, v, L_{1,1}, L_{1,2}, \ldots, L_{1,6}) = (9,1,5,6,4,12,11)$ setzen und das Ergebnis $L' = \big((9,1,5,6,4,12,11),(),(7,2,3,10,8)\big)$ erhalten.

Für die Trennstelle $q = 7$ hätte sich die ursprüngliche Partition L_1 ergeben und das Individuum wäre unverändert geblieben. □

Sei L ein Individuum.

(1) (a) Ziehe gleichverteilt eine Partition $i \in \{1, \ldots, m\}$. Falls $|L_i| = 0$ terminiere.
(b) Ziehe gleichverteilt einen Knoten $p \in \{1, \ldots, |L_i|\}$.

(2) Setze $v \leftarrow L_{i,p}$ und $L_i \leftarrow \big(L_{i,1}, L_{i,2}, \ldots, L_{i,p-1}, L_{i,p+1}, L_{i,p+2}, \ldots, L_{i,|L_i|}\big)$.

(3) (a) Ziehe gleichverteilt eine Partition $k \in \{1, \ldots, m\}$.
(b) Ziehe gleichverteilt eine Trennstelle $q \in \{1, \ldots, |L_k| + 1\}$.

(4) Setze $L_k \leftarrow \big(L_{k,1}, L_{k,2}, \ldots, L_{k,q-1}, v, L_{k,q}, L_{k,q+1}, \ldots, L_{k,|L_k|}\big)$.

Ist $p = 1$ oder $p = |L_i|$ bzw. $q - 1 = 0$ oder $q = |L_k| + 1$, so sind die entsprechenden Teilfolgen in Schritt 2 bzw. Schritt 4 leer.

ALGORITHMUS 4.9: Transposition zwischen zwei Partitionen

Inversionen werden von Algorithmus 4.8 mittels Vertauschungen durchgeführt. Da eine Vertauschung durch zwei Transpositionen ersetzt werden kann, gilt

SATZ 4.5 Jede Inversion kann durch höchstens $4\lfloor n/2 \rfloor$ Transpositionen ersetzt werden. □

BEWEIS Algorithmus 4.8 führt für $p = |L_i| + 1$ und $q = 1$ die maximale Anzahl von

$$\left\lfloor \frac{|L_i| + 1 - 1}{2} \right\rfloor + \left\lfloor \frac{|L_i| + 1 - 1}{2} \right\rfloor = 2 \left\lfloor \frac{|L_i|}{2} \right\rfloor$$

Vertauschungen aus. Die Vertauschung zweier Knoten $L_{h,r} \leftrightarrow L_{j,s}$ kann durch die Vorschrift

(1) Falls $h = j$ und $r = s$ terminiere.

(2) Setze $i \leftarrow h$, $p \leftarrow r$, $k \leftarrow j$ und $q \leftarrow s$.

(3) Führe Algorithmus 4.9 durch.

(4) Setze $i \leftarrow j$, $k \leftarrow h$ und $q \leftarrow r$.
Falls $h = j$ und $r < s$ dann setze $p \leftarrow s - 1$, andernfalls setze $p \leftarrow s + 1$.

(5) Führe Algorithmus 4.9 durch.

erfolgen. Da eine Partition maximal Länge n haben kann, folgt die Behauptung. □

Um festzustellen, wieviele verschiedene Transpositionen für ein Individuum möglich sind, zählen wir die Anzahl der Möglichkeiten für die Parameter i, p, k und q von Algorithmus 4.9. Für den zu verschiebenden Knoten $L_{i,p}$ gibt es n Möglichkeiten, da jeder Knoten verschoben werden kann. Als Einfügestelle kommen nach der Entfernung von $L_{i,p}$ die übrig gebliebenen $n - 1$ Knoten sowie ein Anhängen an jede der m Partitionen in Betracht. Insgesamt ergeben sich $n(n-1+m)$ Möglichkeiten und es gilt

SATZ 4.6 (ANZAHL MÖGLICHER TRANSPOSITIONEN) Auf einem Individuum können $n(m+n-1)$ unterschiedliche Transpositionen ausgeführt werden. □

Mit anderen Worten hat jeder Knoten in dem durch die Transposition definierten Nachbarschaftsgraphen den gleichen Grad $n(m + n - 1) = O(n^2)$ für $m \leq n$. Die Größe der Nachbarschaft eines Knotens wird für die iterierte lokale Suche (siehe Abschnitt 4.4.10.6) von Bedeutung sein.

4.4.10.4 Populationsgesteuerte Inversion und Transposition

Mutationsoperatoren sind, wie bereits zu Beginn von Abschnitt 4.4.10 ausgeführt, in der Regel lokale Operatoren in dem Sinne, dass sie ausschließlich mit dem übergebenen Individuum arbeiten. Guo Tao und Zbigniew Michalewicz [54] haben den auf der Inversion basierenden Operator inver-over für das TSP entwickelt, bei dem die Inversionen nicht rein zufällig gewählt werden, sondern zielgerichtet anhand anderer Individuen aus der Population bestimmt werden.[16] Der von den beiden Autoren auf Grundlage des inver-over-Operators entwickelte Algorithmus 4.10 erzielte mit wenig Rechenzeit gute Ergebnisse auf Instanzen des TSP.[17]

Sei $p \in [0,1] \subset \mathbb{R}$.

(1) Initialisiere die Population mit zufälligen Individuen (Touren).

(2) Solange die Terminierungsbedingung nicht erfüllt ist, führe die folgenden Schritte für jedes Individuum S der Population aus:

  (a) Setze $S' \leftarrow S$.

  (b) Wähle zufällig gleichverteilt eine Stadt c aus S'.

  (c) Mit Wahrscheinlichkeit p

    (i) Wähle c' zufällig aus den verbleibenden Städten von S'.

  andernfalls

    (i) Wähle zufällig gleichverteilt ein Individuum der Population aus.

    (ii) Bestimme c' als den rechten Nachbarn von c in diesem Individuum.

  (d) Falls c' kein Nachbar von c in S' ist

    (i) Invertiere den Abschnitt zwischen c und c', so dass c' rechts neben c zu liegen kommt.

    (ii) Setze $c \leftarrow c'$.

    (iii) Gehe zu Schritt 2c.

  (e) Falls S' einen besseren oder gleich guten Fitnesswert wie S hat, setze $S \leftarrow S'$.

Algorithmus 4.10: Heuristik von Guo Tao und Zbigniew Michalewicz für das TSP auf Grundlage des inver-over-Operators [54]

Wir stellen mit Algorithmus 4.11 eine an das WCP-A angepasste Version vor. Dabei entspricht die Wahl einer Stadt c bzw. c' in Algorithmus 4.10 der Wahl einer Partition i und eines Knotens p bzw. einer Partition k und eines Knotens q in Algorithmus 4.11. Im Unterschied zum TSP kann der Knoten q nicht immer durch eine Inversion zum rechten Nachbarn von p gemacht werden, da die beiden Knoten in unterschiedlichen Partitionen liegen können. In diesem Fall verwenden wir eine Transposition statt einer Inversion. Beispiel 4.7 verdeutlicht den Ablauf.

[16] Der Operator ähnelt dadurch sowohl einem Mutationsoperator, da er nur ein einzelnes Individuum lokal verändert, als auch einem Rekombinationsoperator, da die verwendeten Informationen von mehreren Individuen stammen. Streng genommen handelt es sich weder um einen Mutationsoperator (lokalen Zug), da Informationen anderer Individuen verwendet werden, noch um einen Rekombinationsoperator im eigentlichen Sinne, da das Individuum nur lokal verändert und nicht neu erstellt wird. Wir ordnen den inver-over Operator sowie den auf ihm basierenden Algorithmus 4.11 für das WCP-A aufgrund des Sprachgebrauchs in der Literatur und der Ähnlichkeit mit den Mutationsoperatoren Inversion und Transposition als Mutationsoperator ein.

[17] Für Instanzen bis zu 105 Städten wurde praktisch immer das globale Optimum gefunden; bei größeren Instanzen (144, 442 und 2392 Städte sowie eine zufällige Instanz mit 10 000 Städten) lag die Lösung um weniger als 3% über dem globalen Optimum bzw. der Held-Karp-Schranke. [54]

Seien $I_1, \ldots, I_\mu$ die Individuen der Population, L das übergebene Individuum und $w \in [0,1] \subset \mathbb{R}$ eine Wahrscheinlichkeit.

(1) Setze $L' \leftarrow L$.

(2) (a) Wähle gleichverteilt $i \in \{1, \ldots, m\}$.
(b) Wähle gleichverteilt $p \in \{1, \ldots, |L_i| + 1\}$.

(3) Mit Wahrscheinlichkeit w,

(a) Wähle gleichverteilt $k \in \{1, \ldots, m\}$.
(b) Wähle gleichverteilt $q \in \{1, \ldots, |L_k| + 1\}$.

andernfalls

(c) Wähle gleichverteilt ein Individuum $I' \in I_1, \ldots, I_\mu$.
(d) Falls $p = |L_i| + 1$
(i) Falls $|I'_i| = 0$ gehe zu Schritt 3,
andernfalls setze $k \leftarrow$ Partition von $I'_{i,1}$ in L und $q \leftarrow$ Position von $I'_{i,1}$ in L_k.

andernfalls

(ii) Bestimme Partition h und Position j von $L_{i,p}$ in I'.
(iii) Ist $j = |I'_h|$ dann setze $k \leftarrow h$ und $q \leftarrow |L_k| + 1$,
andernfalls setze $k \leftarrow$ Partition von $I'_{h,j+1}$ in L und $q \leftarrow$ Position von $I'_{h,j+1}$ in L_k.

(4) Falls $L_{k,q}$ bereits ein Nachbar von $L_{i,p}$ ist, gehe zu Schritt 8.

(5) Falls $i \neq k$

(a) Falls $q = |L_k| + 1$, gehe zu Schritt 3.
(b) Setze $p \leftarrow p + 1$. Falls $p = |L_i| + 2$ dann setze $p \leftarrow 1$.
(c) Verwende Algorithmus 4.9 um $L_{k,q}$ nach $L_{i,p}$ zu verschieben.

andernfalls

(d) Setze $p \leftarrow p + 1$. Falls $p = |L_i| + 2$ dann setze $p \leftarrow 1$.
(e) Setze $q \leftarrow q + 1$. Falls $q = |L_i| + 2$ dann setze $q \leftarrow 1$.
(f) Invertiere gemäß Algorithmus 4.8.

(6) Setze $p \leftarrow q$ und $i \leftarrow k$.

(7) Gehe zu Schritt 3.

(8) Falls $f(L') > f(L)$ dann setze $L \leftarrow L'$.

Die Variablen h, j bezeichnen Partition und Position des aktuellen Knotens $L_{i,p}$ in I', die Variablen k, q Partition und Position des rechten Nachbarn von $I'_{h,j}$ in L. Ist p in Schritt 3d der Startknoten s_i, so wird der rechte Nachbar $I'_{i,1}$ des i-ten Startknotens in I' betrachtet. Ist p in Schritt 3diii der letzte Knoten der Partition I'_h, so ist sein rechter Nachbar der Startknoten s_h. In den Schritten 3di und 5a wird die Schleife (Schritte 3 bis 7) vorzeitig beendet, da im ersten Fall $p = s_i$ keinen rechten Nachbarn in I' hat, im andern Fall der Startknoten $q = s_k$ nicht verschoben werden kann. Durch die Transposition bzw. Inversion in Schritt 5 wird $L_{k,q}$ zum rechten Nachbarn von $L_{i,p}$.

ALGORITHMUS 4.11: Populationsgesteuerte Inversion und Transposition

Beispiel 4.7 (Populationsgesteuerte Inversion und Transposition) Es sei die aus den zwei Individuen $I_1 = \big((1,7,3,4,2,6,5),()\big)$ und $I_2 = \big((7,5,3),(6,4,1,2)\big)$ bestehende Population, das Eingabeindividuum $L = \big((4,6,1,3),(5,7,2)\big)$ und die Wahrscheinlichkeit $w = 0{,}02$ gegeben.

Wir sichern zuerst mit $L' \leftarrow L$ in Schritt 1 das Eingabeindividuum. Anschließend wählen wir in Schritt 2 zufällig $i = 2$ und $p = |L_2| + 1 = 4$; in Schritt 3 trete der zweite Fall ein mit $I' = I_1$ in Schritt 3c. Wir fahren wegen $p = |L_2| + 1$ mit Schritt 3di fort; dort ist $|I'_2| = 0$ und wir kehren zu Schritt 3 zurück. Dort trete wieder der zweite Fall ein, diesmal jedoch mit $I' = I_2$. In Schritt 3di ist $|J_2| = 4$ und wir setzen $k \leftarrow$ Partition von $I'_{2,1}$ in $L = 1$ und $q \leftarrow 2$. Da q kein Nachbar von p in L ist und $i = 2 \neq 1 = k$ sowie $q = 2 \neq |L_2| + 1 = 4$, fahren wir mit Schritt 5b fort, setzen $p \leftarrow 1$ und verschieben $L_{1,2}$ gemäß Algorithmus 4.8 an die Trennstelle $L_{i,p}$ mit dem Ergebnis $L = \big((4,1,3),(6,5,7,2)\big)$. Wir setzen $p \leftarrow 2$, $i \leftarrow 1$ und gehen zu Schritt 3.

Dort trete der zweite Fall ein. Mit $I' = 1$, $h \leftarrow 1$, $j \leftarrow 1$, $k \leftarrow 2$ und $q \leftarrow 3$ wird in Schritt 5c Knoten $L_{2,3} = 7$ an die Stelle $L_{1,3}$ verschoben; es ergibt sich $L = \big((4,1,7,3),(6,5,2)\big)$. Wir setzen $p \leftarrow 3$, $i \leftarrow 2$ und kehren zu Schritt 3 zurück.

Es trete der erste Fall ein und wir wählen k, q gleichverteilt mit $k = 2$ und $q = 4$. Da $L_{2,4}$ bereits rechter Nachbar von $L_{2,3}$ ist, terminiert der Algorithmus in Schritt 8.

□

4.4.10.5 Simuliertes Abkühlen

Das simulierte Abkühlen ist ein stochastisches Optimierungsverfahren (siehe Abschnitt 2.3.3.2); man kann es als Spezialfall der evolutionären Algorithmen auffassen. Wir beschreiben einen Mutationsoperator, mit dessen Hilfe wir eine einfache Version des simulierten Abkühlens innerhalb unseres evolutionären Ansatzes nachbilden. Wir gehen in diesem Abschnitt nur auf den Mutationsoperator selbst ein, die Einstellung der anderen Parameter wie z. B. der Populationsgröße wird in Abschnitt 5.4.3 beschrieben.

Das simulierte Abkühlen basiert auf folgender Idee: Ein Individuum wird mittels eines lokalen Zugs modifiziert. Verbessert sich die Fitness, behält man die Modifikation bei. Verschlechtert sich die Fitness, behält man die Modifikation mit einer Wahrscheinlichkeit bei, die von der Größe der Verschlechterung sowie dem zeitlichen Verlauf des Algorithmus abhängt. Dieses Vorgehen wird solange iteriert bis eine Abbruchbedingung erfüllt ist. Für detaillierte Darstellungen verweisen wir auf die Literatur. [46]

Zur Modifikation des Individuums eignen sich die Mutationsoperatoren Inversion und Transposition sowie Kombinationen der beiden. Wir erlauben die Angabe einer Wahrscheinlichkeit $w \in [0{,}1] \subset \mathbb{R}$ und führen mit Wahrscheinlichkeit w eine zufällige Transposition, mit Wahrscheinlichkeit $1-w$ eine zufällige Inversion aus. Bei der Wahl von w ist zu beachten, dass die Transposition im Gegensatz zur Inversion die Erreichbarkeit des gesamten Suchraums gewährleistet.

Bezeichne L' das modifizierte Individuum und sei $\Delta f := f(L') - f(L)$ die durch die Transposition bzw. Inversion bewirkte Veränderung der Fitness. Im Fall $\Delta f \geq 0$ wird L' stets akzeptiert. Die Wahrscheinlichkeit der Akzeptanz von L' im Fall $\Delta f < 0$ ist

$$\left(e^{-\frac{1}{T}}\right)^{-\Delta f} = e^{\frac{\Delta f}{T}}, \quad T \in \mathbb{R}_{>0}.$$

Dabei ist die Temperatur T eine von der Anzahl bereits simulierter Generationen abhängige, monoton fallende Folge positiver Zahlen. Man beachte, dass wegen

$$\lim_{T \to 0} e^{-1/T} = 0 \qquad \text{und} \qquad \lim_{T \to \infty} e^{-1/T} = 1$$

sowie der Monotonie der Exponentialfunktion $e^{-1/T} \in]0,1[\subset \mathbb{R}$ und somit $e^{-\Delta f/-T} \in]0,1[\subset \mathbb{R}$ gilt. Wir verwenden einen exponentiellen Verlaufsplan für die Abkühlung und setzen

$$T(\tau - 1) := T_0 \cdot \alpha^{\tau - 1}, \quad \tau \in \mathbb{N},\ T_0 \in \mathbb{R}_{>0},\ \alpha \in]0,1[\subset \mathbb{R}.$$

Die Starttemperatur $T_0 \in \mathbb{R}_{>0}$ sollte so gewählt werden, dass zu Beginn ($\tau = 1$) alle, zumindest jedoch die Hälfte der Züge akzeptiert werden. Die Starttemperatur ist somit von Δf abhängig:

$$e^{\frac{\Delta f}{T_0 \alpha^{\tau-1}}} \geq \frac{1}{2} \overset{\tau=1}{\iff} \frac{\Delta f}{T_0} \geq \ln\left(\frac{1}{2}\right) \iff \frac{\Delta f}{\ln\left(\frac{1}{2}\right)} \leq T_0 \iff T_0 \geq \frac{-\Delta f}{\ln 2} \in \mathbb{R}_{>0}.$$

Da Δf von der Startlösung und der ersten Transposition bzw. Inversion abhängt, kann T_0 auf diese Weise nicht im Voraus berechnet werden. Wir umgehen dies, indem wir ausnutzen, dass $-\Delta f$ durch die Summe aller Kantengewichte beschränkt ist, und setzen

$$T_0 := \frac{M}{\ln 2} \geq \frac{-\Delta f}{\ln 2} \quad \text{mit} \quad M := \sum_{e \in E} c(e).$$

Die Konstante $\alpha \in]0,1[\subset \mathbb{R}$ kontrolliert die Geschwindigkeit der Abkühlung: Je näher α bei 1 liegt, desto langsamer sinkt T. Wir fassen α als Parameter des Mutationsoperators auf. Für gewöhnlich wird α nahe bei 1 gewählt. Abbildung 4.8 zeigt den Verlauf der Temperatur für verschiedene Werte von α, Algorithmus 4.12 die einzelnen Rechenschritte.

Sei L ein Individuum, $\alpha \in]0,1[\subset \mathbb{R}$ und $w \in [0,1] \subset \mathbb{R}$ eine Wahrscheinlichkeit.

(1) Führe mit Wahrscheinlichkeit w Algorithmus 4.9, andernfalls Algorithmus 4.8 mit L als Eingabe aus. Es bezeichne L' das Ergebnis.

(2) Setze $\Delta f \leftarrow f(L') - f(L)$.

  (a) Fall 1: $\Delta f \geq 0$
  Setze $L \leftarrow L'$.

  (b) Fall 2: $\Delta f < 0$

    (i) Setze $T \leftarrow \frac{\alpha^{\tau-1}}{\ln 2} \cdot \sum_{e \in E} c(e)$.

    (ii) Mit Wahrscheinlichkeit $e^{\frac{\Delta f}{T}}$ setze $L \leftarrow L'$.

Algorithmus 4.12: Simuliertes Abkühlen

4.4.10.6 Iterierte lokale Suche

Eines der erfolgreichsten stochastischen Optimierungsverfahren für das TSP ist die Lin-Kernighan-Heuristik [62], ein Vertreter der iterierten lokalen Suche (siehe Abschnitt 2.3.3.3). Die Idee dieser Metaheuristik ist die Anwendung eines lokalen Optimierungsverfahrens, gelegentlich unterbrochen von größeren zufälligen Modifikationen der Lösung, um lokalen Optima zu entkommen. Eine stochastische Variante dieses Verfahrens ist durch geeignete Parametereinstellungen mit den bereits vorgestellten Bestandteilen des evolutionären Ansatzes möglich (siehe Abschnitt 5.4.4.1). Darüber hinaus stellen wir einen eigens für die iterierte lokale Suche entworfenen Mutationsoperator vor.

Wir verwenden dazu den durch die Transposition definierten Nachbarschaftsgraphen. Da der Transpositionsoperator reversibel ist, ist der Graph ungerichtet. Wir verzichten auf die Hinzunahme von Inversionen, da nach Satz 4.5 jede Inversion durch (höchstens $4\lfloor n/2 \rfloor$) Transpositionen darstellbar ist.

Als lokales Optimierungsverfahren verwenden wir das diskrete Analogon zum Gradientenabstiegsverfahren für kontinuierliche Probleme: Dabei geht man entlang der Kanten des Nachbarschaftsgraphen solange zu Individuen mit höherer[18] Fitness, bis dies nicht mehr möglich ist, d. h. bis

[18] Erlaubt man den Wechsel zu Individuen mit gleicher Fitness, riskiert man, sich im Nachbarschaftsgraphen auf Zyklen zu bewegen.

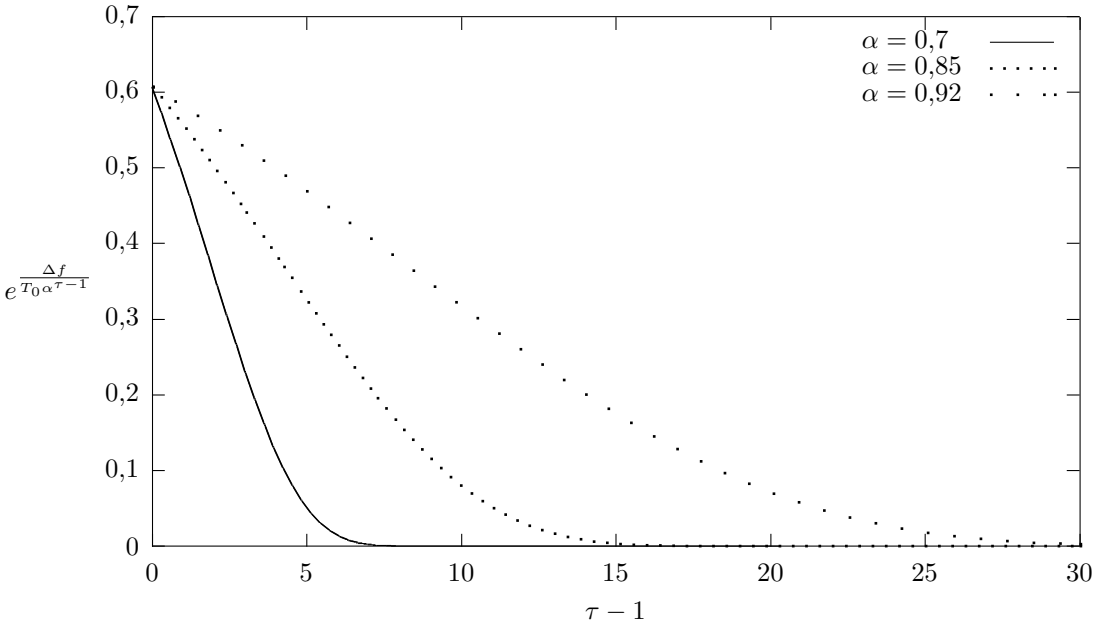


Abbildung 4.8: Abnahme der Akzeptanzwahrscheinlichkeiten beim simulierten Abkühlen für $\Delta f = -0{,}5$, $T_0 = 1$ und $\alpha \in \{0{,}7, 0{,}85, 0{,}92\}$

zu einem Individuum, dessen Fitnesswert besser ist als die Fitnesswerte aller seiner Nachbarn (lokales Optimum). Stehen in einem Schritt mehrere Kanten zur Auswahl, besteht die Möglichkeit, sich entlang des größten Gradienten zu bewegen (engl. steepest descent), indem man den Nachbarn mit dem größten Fitnesswert wählt. Dazu ist es in jedem Schritt erforderlich, alle Kanten eines Knotens zu betrachten. Gemäß Satz 4.6 hat ein Individuum $n(m+n-1) = O\left(n^2 + mn\right)$ Nachbarn. Wir entscheiden uns daher aus Gründen der Laufzeit gegen die Betrachtung aller Nachbarn eines Individuums in jedem Schritt und wählen stets die erste gefundene Verbesserung.

Der Mutationsoperator geht in zwei Schritten vor: Ist das Individuum bereits Ergebnis einer lokalen Suche, werden mehrere zufällige Inversionen bzw. Transpositionen hintereinander ausgeführt. Art und Anzahl dieser Schritte fassen wir als Parameter auf. Anschließend wird das Individuum dem beschriebenen lokalen Optimierungsverfahren unterzogen. Algorithmus 4.13 schildert das genaue Vorgehen.

4.5 Weiterführendes

Der in diesem Kapitel vorgestellte evolutionäre Ansatz bietet Raum für zahlreiche weitere Gestaltungsmöglichkeiten:

Repräsentation Durch einen vom Phänotyp verschiedenen Genotyp lässt sich Redundanz einbringen (Neutrale Mutation, siehe Abschnitt 4.3.2), z. B. in der Form diploider Chromosomensätze. Denkbar sind sowohl dominanzbasierte Ansätze als auch eine Variante des Pseudo-Meiose-Ansatzes (siehe Seite 52).

Nebenbedingungen Um den vorgestellten Ansatz auf das WCP-B zu übertragen, ist die Berücksichtigung von Nebenbedingungen erforderlich. Hierzu existieren verschiedene Ansätze, auf die ihn Abschnitt 4.3.4 eingegangen wird.

Sei $s \in \mathbb{N}_0$, L ein Individuum und $w \in [0,1] \subset \mathbb{R}$ eine Wahrscheinlichkeit.

(1) Falls $\tau - 1 > 0$ dann führe s-mal jeweils mit Wahrscheinlichkeit w Algorithmus 4.9, andernfalls Algorithmus 4.8 auf L aus.

(2) Iteriere i von 1 bis m:

  (a) Falls $|L_i| = 0$, fahre mit Iteration 2 fort.

  (b) Iteriere p von 1 bis $|L_i|$, k von 1 bis m, q von 1 bis $|L_k| + 1$:

    (i) Falls $i = k$ und $q = |L_k| + 1$, fahre mit Iteration 2b fort.

    (ii) Führe Algorithmus 4.9 mit i, p, k, q durch. Es bezeichne L' das Ergebnis.

    (iii) Falls $f(L') > f(L)$, setze $L \leftarrow L'$ und beginne wieder mit Schritt 2.

In Schritt 1 werden für $w = 1$ nur Transpositionen, für $w = 0$ nur Inversionen ausgeführt. Falls in Schritt 2biii ein Individuum mit größerer Fitness gefunden wird, beginnt die Iteration in Schritt 2 wieder mit $i = 1$. Setzt man $s = 0$, führt der Operator nur die lokale Optimierung durch.

Algorithmus 4.13: Iterierte lokale Suche

Diversität Es bietet sich sowohl die Untersuchung der Entwicklung der Diversität in der Population über den Berechnungsverlauf des Algorithmus hinweg an als auch die Kontrolle der Diversität mittels der in Abschnitt 4.3.3 vorgestellten Maßnahmen. Es ist ein Maß für die Diversität der Population zu entwickeln.

Dynamische Methoden Die in diesem Kapitel vorgestellten Bestandteile sind alle statisch in dem Sinne, dass sie über die Zeit (gemessen in simulierten Generationen) hinweg konstant sind, mit Ausnahme der Temperatur im Mutationsoperator für das simulierte Abkühlen. Deren exponentieller Abkühlungsverlauf ist jedoch fest vorgegeben und lässt sich nur indirekt durch die Wahl von α beeinflussen. Die Parameter selbst sind ausnahmslos statisch. Ob ein Parameter optimal ist, kann aber vom zeitlichen Verlauf der Berechnung abhängen: In verschiedenen Stadien des Algorithmus können unterschiedliche Werte für einen Parameter günstiger sein. Diesem Umstand tragen dynamische Parameter besser Rechnung.

Modifikation bestehender Methoden Die vorgestellten Bestandteile des Algorithmus bieten zum Teil noch erhebliches Potential für Erweiterungen; dies gilt insbesondere für die durch unseren Ansatz abgebildeten eigenständigen Optimierungsverfahren wie das simulierte Abkühlen und die iterierte lokale Suche.

Auch die eigens für das WCP-A entworfenen Operatoren lassen sich modifizieren, z. B. kann man in Algorithmus 4.7 zur Kantenrekombination für das WCP-A die Schritte 3bii und 3biii vertauschen; man erhält so eine Variante der Kantenrekombination, die gut fortsetzbare Touren gegenüber solchen bevorzugt, die lokale Teilsequenzen der Eltern erhalten. Eine weitere Variante dieses Rekombinationsoperators verteilt die Knoten in Schritt 1 deterministisch anhand der Häufigkeit der Zuordnungen auf die Partitionen und lässt nur bei gleich häufigen Partitionen den Zufall entscheiden.

Neue Methoden Neben weiteren generischen Methoden zur Initialisierung, Selektion, Rekombination und Mutation von Individuen gibt es zahlreiche, für verwandte Probleme wie TSP, CVRP und JSP entworfene Methoden, die sich auf das WCP-A übertragen lassen.

Auch die Abbildung weiterer Heuristiken wie z. B. der Tabu-Suche (siehe Abschnitt 2.3.3.3) ist denkbar.

Analyse Neben der empirischen Analyse von kompletten Parametersätzen, wie sie im nächsten Kapitel durchgeführt wird, gibt es auch die Möglichkeit, einzelne Teile des Algorithmus empirisch oder formal zu analysieren. Ein Beispiel hierfür ist die ausführliche Untersuchung der Eigenschaften verschiedener Selektionsoperatoren in einem Artikel [52] von Tobias Blickle und Lothar Thiele.

Berücksichtigung von Kollisionen Für praktische Zwecke lässt sich der evolutionäre Algorithmus für das WCP-A modifizieren, indem man alle Lösungskandidaten auf Kollisionsfreiheit prüft. Der Suchraum reduziert sich dadurch in Abhängigkeit von der Probleminstanz.

Kapitel 5

Parametersätze und Analyse

Die im vorherigen Kapitel vorgestellten Bestandteile für einen evolutionären Ansatz lassen sich auf verschiedene Art und Weise miteinander kombinieren. Zusammen mit der Wahl von Kenngrößen wie μ und λ ergibt sich eine Vielzahl möglicher evolutionärer Algorithmen für das WCP-A. Wir stellen einige davon vor und vergleichen sie hinsichtlich der Güte der erbrachten Ergebnisse und der dazu benötigten Rechenleistung. Da konventionelle Analysemethoden wie in Abschnitt 3.2.2 diskutiert der vorliegenden Situation nicht unbedingt gerecht werden und eine klassische Laufzeitanalyse ohnehin nur für sehr einfache Parametersätze möglich wäre, erfolgt der Vergleich auf empirischer Basis.

5.1 Parameter und Abhängigkeiten

Je nachdem, welche der in Abschnitt 4.4 vorgestellten Bestandteile man mit welchen Einstellungen zulässt, ergeben sich unterschiedliche evolutionäre Algorithmen für das WCP-A. Wir fassen den vorgestellten Ansatz als (Meta-) Algorithmus auf, der sich über Kenngrößen sowie seine Bestandteile und deren Einstellungen parametrisieren lässt. Jeder Parametersatz entspricht damit einem evolutionären Algorithmus. Wir geben Parametersätze in der in Abbildung 5.1 gezeigten Form an. Im Folgenden werden die einzelnen Parameter mit zulässigen Werten und Einschränkungen erläutert.

Populationsgrösse (3) (Abschnitt 4.4.4) Entweder als absolute Angabe der Form „$\mu = k$“ mit $k \in \mathbb{N}$ oder als relative Angabe der Form „$\mu' = k'$“ mit $k' \in \mathbb{R}_{>0}$.

Zu Einschränkungen siehe „Anzahl der Kinder (4)“.

Anzahl der Kinder (4) (Abschnitt 4.4.4) Entweder als absolute Angabe der Form „$\lambda = k$“ mit $k \in \mathbb{N}$ oder als relative Angabe der Form „$\lambda' = k'$“ mit $k' \in \mathbb{R}_{>0}$.

Es gelten die Einschränkungen aus Tabelle 4.2: Ist die Selektionsstrategie (5) eine Komma-Strategie (μ, λ) und soll die Anzahl der Kinder durch eine relative Angabe spezifiziert werden, muss $\lambda' \geq 1$ gelten; soll die Anzahl der Kinder durch eine absolute Angabe spezifiziert werden, muss die Populationsgröße (3) ebenfalls durch eine absolute Angabe spezifiziert sein und es muss $\lambda \geq \mu$ gelten.

Selektionsstrategie (5) (Abschnitt 4.4.8.2) Entweder von der Form „(μ, λ)“ für eine Komma-Selektionsstrategie oder von der Form „$(\mu + \lambda)$“ für eine Plus-Selektionsstrategie.

Parameter	A	R	Wert
Name			(1) ((2))
Populationsgröße			(3)
Anzahl der Kinder			(4)
Selektionsstrategie			(5)
Terminierung			(6)
Initialisierung	(7)	(8)	(9)
Paarungsselektion	(10)	(11)	(12)
Umweltselektion	(13)	(14)	(15)
Rekombination	(16)	(17)	(18)
Mutation	(19)	(20)	(21)

(1) Name des Algorithmus; kurze Bezeichnung, die in Text, Grafiken und Tabellen verwendet wird
(2) Kurze Beschreibung; ein erklärender Name oder ein kurzer Kommentar
(3)–(21) Erläuterungen im Fließtext ab Seite 83

Tabelle 5.1: Angabe von Parametersätzen

Terminierung (6) (Abschnitt 4.4.7) Zulässige Werte sind

- „Maximale Anzahl an Generationen ($\tau_{\max} = k$)“ mit $k \in \mathbb{N}_0$
- „Keine Verbesserung der besten Individuen ($\tau_{\text{last}} = k$)“ mit $k \in \mathbb{N} \setminus \{1\}$
- „Max. Anzahl an Auswertungen der Fitnessfunktion ($f_{\max} = k$)“ mit $k \in \mathbb{N}_0$

Es muss mindestens eine Terminierungsbedingung angegeben werden.

Initialisierung (7), (8), (9) (Abschnitt 4.4.5) Für jede Methode (9) zur Initialisierung der Population muss eine absolute Anforderung (7) der Form „a_i“ mit $a_i \in \mathbb{N}_0$ und eine relative Anforderung (8) der Form „r_i“ mit $r_i \in \mathbb{N}_0$ angegeben werden. Für diese Anforderungen gelten die Einschränkungen aus Algorithmus 4.2: Ist $k \in \mathbb{N}$ die Anzahl der angegebenen Initialisierungsmethoden, muss $\sum_{i=1}^{k} a_i \leq \mu$ und $\sum_{i=1}^{k} r_i > 0$ gelten.

Zulässige Werte für die Methoden (9) zur Initialisierung sind

- „Uniforme Verteilung“
- „Nächster Roboter & gierige Ersparnis-Heuristik (probabilistisch)“
- „Nächster Roboter & gierige Ersparnis-Heuristik (deterministisch)“

Es muss mindestens eine Methode zur Initialisierung angegeben werden.

Paarungsselektion (10), (11), (12) (Abschnitt 4.4.8.1) Für jede Methode (12) zur Paarungsselektion muss eine absolute Anforderung (10) der Form „a_i“ mit $a_i \in \mathbb{N}_0$ und eine relative Anforderung (11) der Form „r_i“ mit $r_i \in \mathbb{N}_0$ angegeben werden. Für diese Anforderungen gelten die Einschränkungen aus Algorithmus 4.2: Ist $k \in \mathbb{N}$ die Anzahl der Methoden zur Paarungsselektion, muss $\sum_{i=1}^{k} a_i \leq \lambda\, a$ und $\sum_{i=1}^{k} r_i > 0$ gelten. Dabei ist a das Minimum der Anzahlen der benötigten Elternindividuen der Methoden (18) zur Rekombination.[1]

[1] Die tatsächlich benötigte Anzahl an Elternindividuen wird in jeder Generation gemäß Algorithmus 4.5 neu bestimmt, die obere Schranke für die Summe $\sum_{i=1}^{k} a_i$ der absoluten Anforderungen der Methoden zur Paarungsselektion dagegen ist statisch und darf daher nicht größer sein als die minimale Anzahl an Individuen, die von den Rekombinationsoperatoren benötigt werden können. Tatsächlich ist die gewählte Schranke λa nicht scharf, da sie die absoluten Anforderungen der Rekombinationsoperatoren nicht berücksichtigt; für unsere Zwecke ist sie jedoch ausreichend.

Zulässige Werte für die Methoden (12) zur Paarungsselektion sind

- „Deterministische Selektion des besten Individuums“
- „Uniforme Selektion“
- „Lineare rangbasierte Selektion ($c = x$)“ mit $x \in \mathbb{R}_{\geq 0}$
- „Exponentielle rangbasierte Selektion ($c = x$)“ mit $x \in [0, 1[\subset \mathbb{R}_{\geq 0}$
- „Stochastische Turnierselektion ($q = k$)“ mit $k \in \mathbb{N}$

Es muss mindestens eine Methode zur Paarungsselektion angegeben werden.

Umweltselektion (13), (14), (15) (Abschnitt 4.4.8.2) Für jede Methode (15) zur Umweltselektion muss eine absolute Anforderung (13) der Form „a_i“ mit $a_i \in \mathbb{N}_0$ und eine relative Anforderung (14) der Form „r_i“ mit $r_i \in \mathbb{N}_0$ angegeben werden. Für diese Anforderungen gelten die Einschränkungen aus Algorithmus 4.2: Ist $k \in \mathbb{N}$ die Anzahl der Methoden (15) zur Umweltselektion, muss $\sum_{i=1}^{k} a_i \leq \mu$ und $\sum_{i=1}^{k} r_i > 0$ gelten.

Die zulässigen Werte für die Methoden (15) zur Umweltselektion sind die gleichen wie für die Methoden (12) zur Paarungsselektion.

Es muss mindestens eine Methode zur Umweltselektion angegeben werden.

Rekombination (16), (17), (18) (Abschnitt 4.4.9) Für jede Methode (18) zur Rekombination muss eine absolute Anforderung (16) der Form „a_i“ mit $a_i \in \mathbb{N}_0$ und eine relative Anforderung (17) der Form „r_i“ mit $r_i \in \mathbb{N}_0$ angegeben werden. Für diese Anforderungen gelten die Einschränkungen aus Algorithmus 4.2: Ist $k \in \mathbb{N}$ die Anzahl der Methoden (18) zur Rekombination, muss $\sum_{i=1}^{k} a_i \leq \lambda$ und $\sum_{i=1}^{k} r_i > 0$ gelten.

Zulässige Werte für die Methoden (18) zur Rekombination sind

- „Identität“
- „Kantenrekombination ($p = k$)“ mit $k \in \mathbb{N}$

Es muss mindestens eine Methode zur Rekombination angegeben werden.

Mutation (19), (20), (21) (Abschnitt 4.4.10) Für jede Methode (21) zur Mutation muss eine absolute Anforderung (19) der Form „a_i“ mit $a_i \in \mathbb{N}_0$ und eine relative Anforderung (20) der Form „r_i“ mit $r_i \in \mathbb{N}_0$ angegeben werden. Für diese Anforderungen gelten die Einschränkungen aus Algorithmus 4.2: Ist $k \in \mathbb{N}$ die Anzahl der Methoden (21) zur Mutation, muss $\sum_{i=1}^{k} a_i \leq \lambda$ und $\sum_{i=1}^{k} r_i > 0$ gelten.

Zulässige Werte für die Methoden (21) zur Mutation sind

- „Identität“
- „Inversion“
- „Transposition“
- „Populationsgesteuerte Inversion und Transposition ($w = x$)“ mit $x \in [0, 1] \subset \mathbb{R}$
- „Simuliertes Abkühlen ($\alpha = x, w = y$)“ mit $x \in]0, 1[\subset \mathbb{R}$ und $y \in [0, 1] \subset \mathbb{R}$
- „Iterierte lokale Suche ($s = k, w = x$)“ mit $k \in \mathbb{N}_0$ und $x \in [0, 1] \subset \mathbb{R}$

Es muss mindestens eine Methode zur Mutation angegeben werden.

Man beachte, dass die relativen Anforderungen innerhalb einer Gruppe von Methoden bei manchen, aber nicht bei allen Einträgen Null sein dürfen (mindestens ein Eintrag muss von Null verschieden sein).

5.2 Datensätze

Zur empirischen Analyse verwenden wir als Datensätze elf Probleminstanzen, die auf realen Fertigungsdaten beruhen sowie achtzehn zufällig erzeugte Probleminstanzen. Tabelle 5.2 gibt einen Überblick über die Datensätze.

Zur Erstellung der Produktionsdatensätze wurden die euklidischen Abstände zwischen den Schweißpunkten bzw. den Fußpunkten der Roboter (siehe Abschnitt 1.4.2) verwendet. Die zufällig erzeugten Instanzen wurden generiert, indem jedem Knoten und jedem Startknoten ein zufälliger, uniform verteilter Punkt aus $[0, 1[^3 \subset \mathbb{R}^3$ zugewiesen und die Kantengewichte auf die euklidischen Abstände gesetzt wurden.

Kürzel	m	n
A	4	107
B	4	194
C	2	32
D	2	43
E	2	67
F	2	9
G	2	9
H	4	16
I	4	19
J	2	50
K	2	23

(a) Produktionsdaten

Kürzel	m	n
a	2	10
b	2	50
c	3	50
d	4	50
e	2	100
f	3	100
g	4	100
h	6	100
i	8	100
j	3	200
k	4	200
l	6	200
m	8	200
n	12	200
o	6	500
p	8	500
q	12	500
r	12	1000

(b) Zufällige Datensätze

Tabelle 5.2: Datensätze für das WCP-A. Jedem Datensatz ist ein einbuchstabiges Kürzel zugeordnet, mittels dessen im Rest des Kapitels auf ihn verwiesen wird. Produktionsdatensätze haben Großbuchstaben als Kürzel, zufällig erzeugte Datensätze Kleinbuchstaben.

Bei den Produktionsdatensätzen handelt es sich überwiegend um Bodenbleche und Seitenteile von Personenkraftfahrzeugen. Einige der Datensätze sind teilweise redundant, z. B. entspricht Datensatz B dem um Umfahrpunkte (siehe Abschnitt 3.1.5) erweiterten Datensatz A. Die Datensätze F und G sind ähnlich, aber nicht identisch, ebenso die Datensätze H und I. In Anbetracht der knappen Datenlage haben wir uns für die Berücksichtigung aller, auch ähnlicher, Datensätze entschieden, die uns zur Verfügung standen.

5.3 Auswertung

Da wir mit stochastischen Verfahren arbeiten, sind für aussagekräftige experimentelle Ergebnisse mehrere Läufe eines Algorithmus notwendig. Uns stehen dabei für jede Ausführung eines Algorithmus auf einem Datensatz die folgenden Informationen zur Verfügung: Die gefundene Lösung L (das beste Individuum in allen Generationen), die Anzahl der simulierten Generationen $\tau - 1$ sowie für jede Generation die Werte $f_{\max}$, f_{best} und f_{eval} (siehe Abschnitte 4.4 und 4.4.6).

Für jeden der im folgenden Abschnitt 5.4 definierten dreizehn Parametersätze wurden, soweit dies möglich war,[2] 50 Programmläufe auf jedem Datensatz ausgeführt (siehe Abschnitt 5.5.1). Zusammen wurden 18 627 Programmläufe mit insgesamt 140 158 075 simulierten Generationen[3] durchgeführt. Wir erläutern im Folgenden die zur Auswertung dieser Daten verwendeten Methoden und Darstellungen. Auswertungen, die sich auf die Ergebnisse aller Programmläufe beziehen, befinden sich in Abschnitt 5.5 und werden dort beschrieben. Statistische Kenngrößen und ihre Beschreibung finden sich in Anhang A.

Die Güte einer Lösung entspricht ihrer Gesamtbearbeitungszeit, d. h. der Länge der längsten Tour; die Güte eines Individuums entspricht diesem Wert versehen mit einem negativen Vorzeichen, da die Gesamtbearbeitungszeiten zu minimieren, die Fitnesswerte dagegen zu maximieren sind. Innerhalb dieses Kapitels und in Anhang A arbeiten wir durchgängig mit den Gesamtbearbeitungszeiten, d. h. die Ergebnisse sind positive reelle Zahlen, wobei kleinere Zahlen besseren Ergebnissen entsprechen. Dies gilt auch für die Darstellung der Ergebnisse der Algorithmen, d. h. in Abbildungen und Tabellen wird $-f_{\max}$ statt $f_{\max}$ verwendet.

5.3.1 Kriterien

Zum Vergleich der Parametersätze benötigen wir Kriterien, anhand derer der Vergleich erfolgen kann. Für unsere Situation kommen u. a. folgende Kriterien in Betracht:

Ergebnisgüte Der Wert $-f_{\text{best}}$ der gefundenen Lösung L.

Rechenzeit Die Anzahl der benötigten Rechenschritte. Ein grobes Maß hierfür ist f_{eval}.

Effizienz Das Verhältnis von Ergebnisgüte und dem zur Ermittlung des Ergebnisses benötigten Rechenaufwand.

Zuverlässigkeit Im Sinne einer niedrigen Varianz; der Wert der in einem Programmlauf gefundenen Lösung wird dabei als Zufallsvariable aufgefasst.

Da in unserem Szenario die Optimierungsläufe selbst nicht zeitkritisch sind und die Güte der Ergebnisse im Vordergrund steht, entscheiden wir uns für die Ergebnisgüte als Vergleichskriterium. Aussagen über die Effizienz einzelner Algorithmen lassen sich anhand der in Abschnitt 5.3.3.3 beschriebenen Tabellen treffen. Angaben zur Zuverlässigkeit entnimmt man den Tabellen mit den statistischen Kenngrößen der Algorithmen in Anhang A.

5.3.2 Evaluation eines Algorithmus

Wir beschreiben eine grafische Darstellung für Berechnungsverläufe eines Algorithmus. Statistische Kenngrößen wie Schätzer für Erwartungswert und Varianz der Ergebnisse finden sich in Anhang A.

[2] Für Algorithmus A_1 (Parametersatz 5.1) auf Datensatz r und für Algorithmus E_1 (Parametersatz 5.13) auf den Datensätzen o, p, q, r wurden aufgrund der langen Laufzeiten und der begrenzten verfügbaren Rechenzeit weniger bzw. keine Programmläufe durchgeführt (siehe auch Abschnitte 5.4.1, 5.4.5 und 5.5.1).

[3] Die Initialisierung einer Population, die auch als nullte Generation angesehen werden kann, wurde bei der Zählung nicht berücksichtigt.

5.3.2.1 Grafische Darstellung

Die einfachste Art, den Berechnungsverlauf einer einzigen Ausführung eines Algorithmus auf einem Datensatz grafisch darzustellen, ist $-f^{(k)}_{\max}$ über die simulierten Generationen $k \in \{0, \ldots, \tau - 1\}$ hinweg aufzutragen. Wir beschriften dazu die horizontale Achse mit der Anzahl der simulierten Generationen und die vertikale Achse mit der Güte der besten Lösung innerhalb einer Generation.

Da der Verlauf einer einzigen Ausführung eines stochastischen Verfahrens wenig aussagekräftig ist, werten wir alle verfügbaren Läufe des Algorithmus auf diesem Datensatz aus, indem wir statt einem $-f^{(k)}_{\max}$-Wert das Minimum, das Maximum und den Durchschnitt aller für die Generation $k \in \mathbb{N}_0$ verfügbaren $-f^{(k)}_{\max}$-Werte darstellen. Sei $\mathfrak{f}^{(k)} \in \mathbb{R}^{\mathbb{N}}$ der Vektor mit allen $-f^{(k)}_{\max}$-Werten, die für Generation k zur Verfügung stehen. Wir setzen

$$\mathfrak{f}^{(k)}_{\min} := \min\left\{ \mathfrak{f}^{(k)}_i \;\middle|\; 1 \leq i \leq \left|\mathfrak{f}^{(k)}\right| \right\},$$
$$\mathfrak{f}^{(k)}_{\max} := \max\left\{ \mathfrak{f}^{(k)}_i \;\middle|\; 1 \leq i \leq \left|\mathfrak{f}^{(k)}\right| \right\},$$
$$\mathfrak{f}^{(k)}_{\varnothing} := \frac{1}{\left|\mathfrak{f}^{(k)}\right|} \sum_{i=1}^{\left|\mathfrak{f}^{(k)}\right|} \mathfrak{f}^{(k)}_i$$

und zeichnen diese Größen für k zwischen Null und der Länge des längsten Programmlaufs. Die Größe $\mathfrak{f}^{(k)}_{\varnothing}$ ist ein erwartungstreuer Schätzer für den Erwartungswert $\mathbb{E}[-f_{\max}]$ in der k-ten Generation des Algorithmus. Man beachte, dass die Güte dieses Schätzers mit der Anzahl der simulierten Generationen monoton abnimmt, da immer mehr Algorithmen terminieren und so weniger Stichproben für eine Generation zur Verfügung stehen. Bedingt durch die Abnahme der verfügbaren $-f^{(k)}_{\max}$-Werte kann es zu Sprungstellen im Kurvenverlauf kommen. Um die Güte des Schätzers und damit die sich über seinen Verlauf hinweg ändernde Aussagekraft der Abbildung einschätzen zu können, tragen wir die Anzahl der Programmläufe, die zu einer Generation beigetragen haben, am oberen Rand der Abbildung ab.

Die beschriebene Abbildung gibt einen optischen Eindruck vom Verlauf der Konvergenz der Berechnung und von der Spannbreite der Ergebnisse (Minimum, Maximum, Schwerpunkt) des Algorithmus auf dem ausgewählten Datensatz. Ein Beispiel für diese Art der Darstellung ist die Abbildung 5.1.

5.3.3 Vergleich mehrerer Algorithmen

Wir beschreiben eine grafische Darstellung der Berechnungsverläufe mehrerer Algorithmen auf einem Datensatz, eine Methode für den direkten Vergleich der Ergebnisse zweier Algorithmen auf mehreren Datensätzen und Tabellen für die Gegenüberstellung der Anzahl der Auswertungen der Fitnessfunktion zweier Algorithmen auf mehreren Datensätzen.

5.3.3.1 Grafische Darstellung

Zur vergleichenden Darstellung der Ergebnisse mehrerer Algorithmen auf dem gleichen Datensatz berechnen wir für jeden Algorithmus $\mathfrak{f}^{(k)}_{\varnothing}$ wie in Abschnitt 5.3.2.1 und tragen die entsprechenden Kurven in die Abbildung ein. Auf $\mathfrak{f}^{(k)}_{\min}$ und $\mathfrak{f}^{(k)}_{\max}$ verzichten wir zu Gunsten der Übersichtlichkeit.

Man beachte, dass wie in Abschnitt 5.3.2.1 die Aussagekraft der eingezeichneten Kurven mit abnehmender Anzahl der beitragenden Berechnungsverläufe geringer wird. Um die Aussagekraft der Kurven besser einschätzen zu können, tragen wir für jeden Algorithmus die Anzahl der Programmläufe, die zu einer Generation beigetragen haben, am oberen Rand der Abbildung in der Form „$x/y/\ldots$“ ab. Dabei steht x für die Anzahl der zu dieser Generation beitragenden Programmläufe des ersten Algorithmus, y für die des zweiten Algorithmus usw.

Beispiele für diese Art der Darstellung sind die Abbildungen 5.2, 5.3, 5.4 und 5.5.

5.3.3.2 Direkter Vergleich

Die grafische Darstellung der gemittelten Ergebnisse eines oder mehrerer Algorithmen auf verschiedenen Datensätzen ist auch dann nicht ratsam, wenn man die Ergebnisse auf ein einheitliches Intervall skaliert, da Länge und Art der Verlaufskurve vom Datensatz abhängen. Um dennoch einen übersichtlichen, direkten Vergleich der Ergebnisse zweier Algorithmen auf mehreren Datensätzen zu ermöglichen, verwenden wir den folgenden Tabellentyp:

Die Zeilen beschriften wir mit den Datensätzen, auf denen die Algorithmen verglichen werden. In einer breiten Tabellenspalte tragen wir von der Mitte ausgehend für jeden Datensatz einen Balken ab, dessen Länge der Differenz der skalierten gemittelten Ergebniswerte der beiden Algorithmen auf diesem Datensatz entspricht. Die Richtung des Balkens zeigt den Algorithmus mit dem besseren Ergebniswert an, die Länge des Balkens ist proportional zum Unterschied in der Ergebnisgüte. Jeder Balken ist zusätzlich mit seiner Länge beschriftet. In der untersten Tabellenzeile geben wir den Durchschnitt dieser Werte über alle aufgeführten Datensätze an.

Die Skalierung erfolgt wie in Abschnitt 5.5.2 beschrieben anhand der unteren Schranken für die Datensätze. Das skalierte Ergebnis eines Algorithmus gibt an, um wieviel Prozent dieses Ergebnis über der unteren Schranke für den entsprechenden Datensatz liegt. Die Beschriftungen der Balken zeigen somit, um wieviel Prozent (bezogen auf die untere Schranke für diesen Datensatz) sich die Ergebnisse der beiden Algorithmen auf diesem Datensatz im Mittel unterscheiden.

Wenn sich die beiden Algorithmen um 100% des Werts der unteren Schranke für einen Datensatz unterscheiden, nimmt der Balken den gesamten ihm zur Verfügung stehenden Raum ein. Ist die Differenz größer, steht für einen Balken proportionaler Länge nicht genügend Raum zur Verfügung. In diesen Fällen verwenden wir Fortsetzungsbalken ←□ bzw. □→. Bei mehreren Fortsetzungsbalken entsprechen ihre Längen der Reihenfolge der Differenzwerte.[4] Die exakte Differenz kann weiterhin dem in der Tabellenzeile eingetragenen numerischen Wert entnommen werden.

Unterscheiden sich die gemittelten Ergebnisse der beiden Algorithmen auf einem Datensatz nicht, d. h. beträgt die Differenz Null, entfällt der Balken und beide Enden der Tabellenzeile werden mit 0,0000 beschriftet.

Beispiele für direkte Vergleiche der Ergebnisgüte sind die Tabellen 5.3, 5.4, 5.5, 5.6, 5.8, 5.9, 5.11, 5.13, 5.14, 5.15 und 5.16.

5.3.3.3 Gegenüberstellung der Anzahl an Auswertungen der Fitnessfunktion

Wir interessieren uns nicht nur für die Güte der erbrachten Ergebnisse, sondern auch für den zur Berechnung dieser Ergebnisse benötigten Rechenaufwand. Als grobes Maß hierfür kann die Anzahl der Auswertungen der Fitnessfunktion dienen.

Wir stellen die Anzahl der Auswertungen der Fitnessfunktion zweier Algorithmen einander in tabellarischer Form gegenüber. Dazu beschriften wir die Zeilen mit den Datensätzen und geben in zwei Spalten die Anzahl der Funktionsauswertungen an, gemittelt über alle auf dem entsprechenden Datensatz durchgeführten Programmläufe. In zwei weitere Spalten tragen wir die Quotienten dieser beiden Zahlen ein; diese können als Prozentwerte interpretiert werden („Algorithmus x benötigt y% mehr Funktionsauswertungen als Algorithmus z“).

Man beachte, dass es sich bei den ersten beiden Tabellenspalten um gerundete Werte handelt. Beispiele für derartige Gegenüberstellungen der Anzahl von Funktionsauswertungen sind die Tabellen 5.7, 5.10 und 5.12.

[4] Die Länge ist bei den Fortsetzungsbalken nicht mehr proportional zum Differenzwert, sondern den Balken werden aufsteigende Längen zugewiesen, wobei die kleinste Differenz den kleinsten Balken erhält, die nächstgrößere Differenz den nächstgrößeren Balken usw. Ist der in der untersten Tabellenzeile eingetragene Durchschnitt der Differenzen größer als 100%, wird dies durch einen Fortsetzungsbalken fester Länge angezeigt.

5.3.4 Interpretation

Bei der in den Abschnitten 5.3.2.1 und 5.3.3.1 beschriebenen grafischen Darstellung von Berechnungsverläufen ist zu beachten, dass der Durchschnitt der Ergebnisse für eine Generation $k \in \mathbb{N}_0$ nur über diejenigen Berechnungsverläufe des Algorithmus gebildet wird, in denen der Algorithmus in Generation k noch nicht terminiert hat. Weiter ist zu berücksichtigen, dass für die Darstellung $-f_{\text{max}}^{(k)}$-Werte und nicht $-f_{\text{best}}^{(k)}$-Werte verwendet werden.

Die Ergebnisse der Algorithmen sind $-f_{\text{best}}^{(\tau-1)}$-Werte, wobei $\tau - 1$ vom Verlauf der Berechnung abhängt. Würde man die Ergebnisse als Punkte $\left(\tau - 1, -f_{\text{best}}^{(\tau-1)}\right)$ in die Darstellung einzeichnen, wären sie um den Kurvenverlauf von $\mathfrak{f}_{\text{max}}^{(k)}$ herum verteilt. Die grafischen Darstellungen können daher nur einen optischen Eindruck vom Verlauf der Berechnungen bieten und sind für quantitative Aussagen über die Güte der Ergebnisse nicht geeignet. Für numerische Aussagen sind die in Abschnitt 5.3.3.2 beschriebenen direkten Vergleiche und die Gesamtauswertung in Abschnitt 5.5 heranzuziehen.

5.4 Parametersätze

Im Folgenden definieren wir verschiedene Parametersätze und analysieren ihr Verhalten auf den Datensätzen. Um sicherzustellen, dass die Algorithmen terminieren, enthalten alle Parametersätze die Terminierungsbedingung „Max. Anzahl an Auswertungen der Fitnessfunktion ($f_{\text{max}} = 10000000$)“. Dies verbessert in geringem Maße auch die Vergleichbarkeit der Algorithmen, da für jeden nur eine begrenzte Menge an Rechenzeit zur Verfügung steht.

5.4.1 Evolutionäre Algorithmen

Wir beginnen mit einem Parametersatz, der einem klassischen evolutionären Algorithmus entspricht. Bei der Populationsgröße setzen wir $\mu' = 0{,}1$ und erhalten ein Individuum für je zehn Schweißpunkte; durch $\lambda' = 1{,}25$ erhalten wir eine gegenüber den Eltern um 25% größere Anzahl an Kindern. Um eine möglichst große Flexibilität bei der Umweltselektion zu erreichen, entscheiden wir uns für die $(\mu + \lambda)$-Selektion. Der Algorithmus terminiert nach 100 Generationen ohne Verbesserung. Wir beginnen mit uniform verteilten Individuen als Startlösungen. Die Paarungsselektion erfolgt durch lineare rangbasierte Selektion mit $c = 2$, was für einen deutlichen, aber nicht allzu hohen Selektionsdruck an dieser Stelle sorgt. Für die Umweltselektion verwenden wir die stochastische Turnierselektion mit $q = 5$ (siehe Abbildung 4.5). Bei der Rekombination kommen zu gleichen Teilen die Identität und die Kantenrekombination zum Einsatz. Die Verwendung der Identität als Rekombinationsoperator sorgt dafür, dass ein Teil der Individuen nur durch die Mutationsoperatoren verändert wird und ermöglicht so kleine Veränderungen der Individuen. Die Kantenrekombination verwendet mit $p = 5$ relativ viele Elternindividuen, um eine gute Fortsetzbarkeit der Teilsequenzen der Eltern zu ermöglichen. Als Mutationsoperatoren verwenden wir Identität, Inversion und Transposition. Tabelle 5.1 fasst die Parameter zusammen.

Um einen ersten Eindruck vom Verhalten von Algorithmus A_1 zu gewinnen, betrachten wir den Berechnungsverlauf auf zwei willkürlich ausgewählten Datensätzen. Abbildung 5.1 zeigt 50 Berechnungsverläufe von Algorithmus A_1 auf dem Produktionsdatensatz E und auf dem zufälligen Datensatz f. Konvergenzverhalten und -geschwindigkeit sind bei beiden Datensätzen sehr ähnlich, wobei die Spannbreite zwischen schlechtestem und bestem Ergebnis innerhalb der ersten 1500 Generationen auf dem zufällig erstellten Datensatz geringer ist.

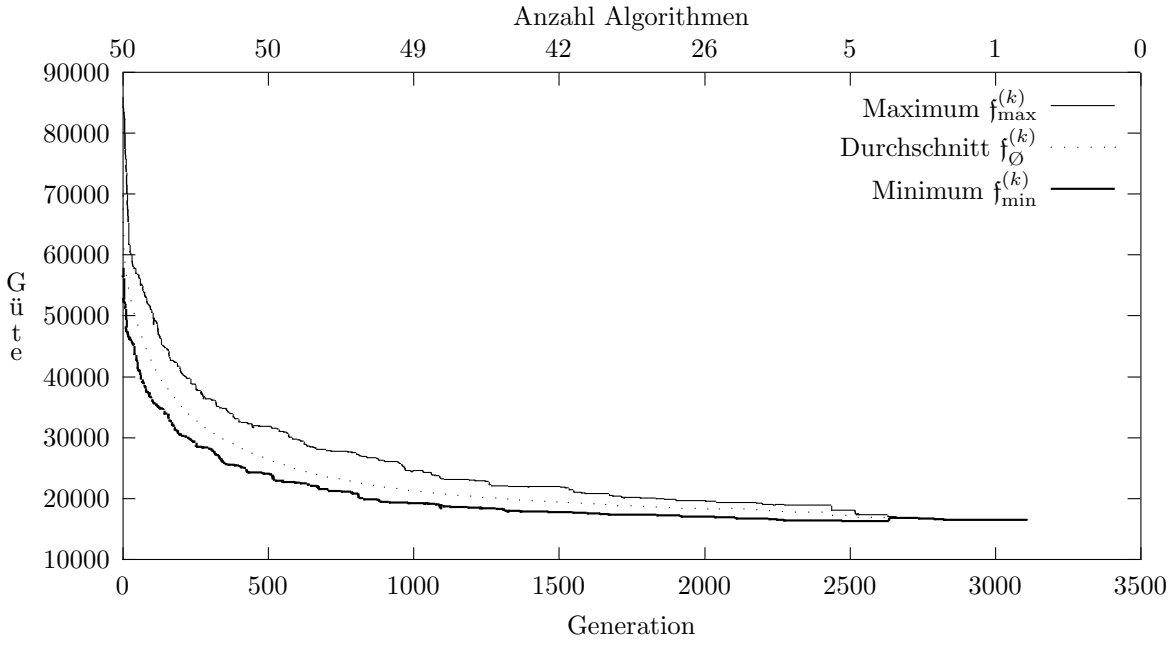
In beiden Fällen beginnt der Algorithmus, bedingt durch die zufälligen Startlösungen, mit sehr schlechten Ergebnissen. Wir ersetzen daher die Initialisierung der Population durch die Heuristik aus Abschnitt 4.4.5.2 und vergleichen die Berechnungsverläufe. Parametersatz 5.2 zeigt die entsprechenden Einstellungen für Algorithmus A_2, Abbildung 5.2 den Berechnungsverlauf der beiden Algorithmen auf dem Produktionsdatensatz E und auf dem zufälligen Datensatz f.

Parameter	A	R	Wert
Name			A_1 (evolutionärer Algorithmus, Variante 1)
Populationsgröße			$\mu' = 0{,}1$
Anzahl der Kinder			$\lambda' = 1{,}25$
Selektionsstrategie			$(\mu + \lambda)$
Terminierung			Keine Verbesserung der besten Individuen ($\tau_{\text{last}} = 100$)
			Max. Anzahl an Auswertungen der Fitnessfunktion ($f_{\max} = 10000000$)
Initialisierung		1	Uniforme Verteilung
Paarungsselektion		1	Lineare rangbasierte Selektion ($c = 2$)
Umweltselektion		1	Stochastische Turnierselektion ($q = 5$)
Rekombination		1	Identität
		1	Kantenrekombination ($p = 5$)
Mutation		1	Identität
		1	Inversion
		1	Transposition

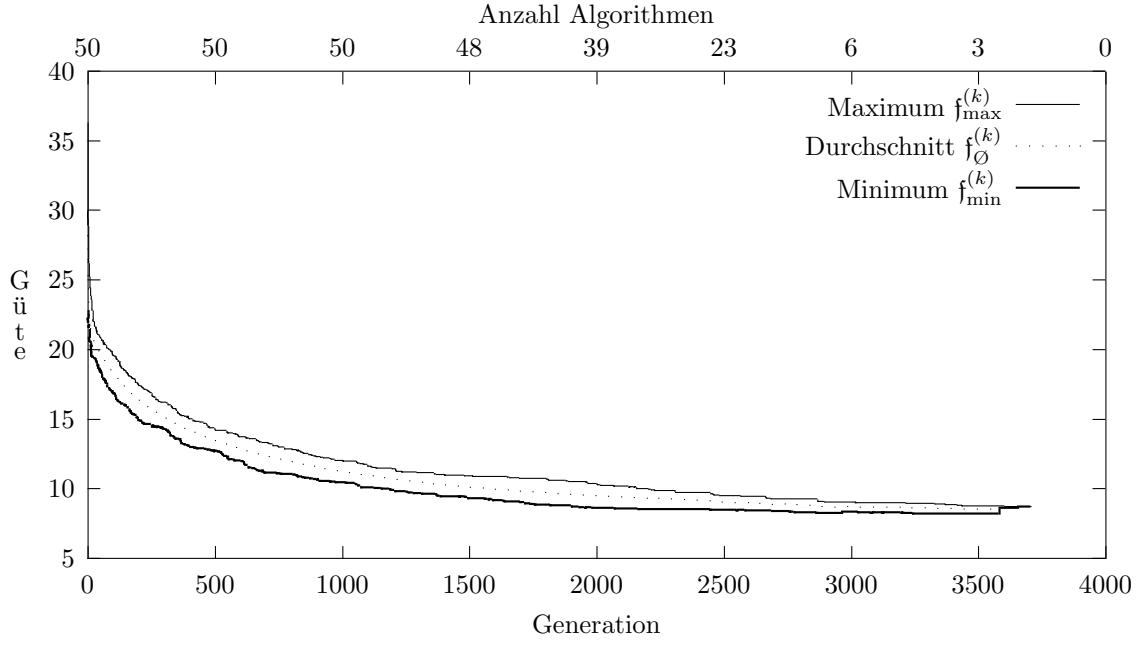
PARAMETERSATZ 5.1: Algorithmus A_1 (evolutionärer Algorithmus, Variante 1)

Parameter	A	R	Wert
Name			A_2 (evolutionärer Algorithmus, Variante 2)
Populationsgröße			$\mu' = 0{,}1$
Anzahl der Kinder			$\lambda' = 1{,}25$
Selektionsstrategie			$(\mu + \lambda)$
Terminierung			Keine Verbesserung der besten Individuen ($\tau_{\text{last}} = 100$)
			Max. Anzahl an Auswertungen der Fitnessfunktion ($f_{\max} = 10000000$)
Initialisierung		1	Nächster Roboter & gierige Ersparnis-Heuristik (probabilistisch)
Paarungsselektion		1	Lineare rangbasierte Selektion ($c = 2$)
Umweltselektion		1	Stochastische Turnierselektion ($q = 5$)
Rekombination		1	Identität
		1	Kantenrekombination ($p = 5$)
Mutation		1	Identität
		1	Inversion
		1	Transposition

PARAMETERSATZ 5.2: Algorithmus A_2 (evolutionärer Algorithmus, Variante 2). Gegenüber Algorithmus A_1 wurde die uniforme Initialisierung durch die nächster Roboter & gierige Ersparnis-Heuristik in der probabilistischen Variante ersetzt.

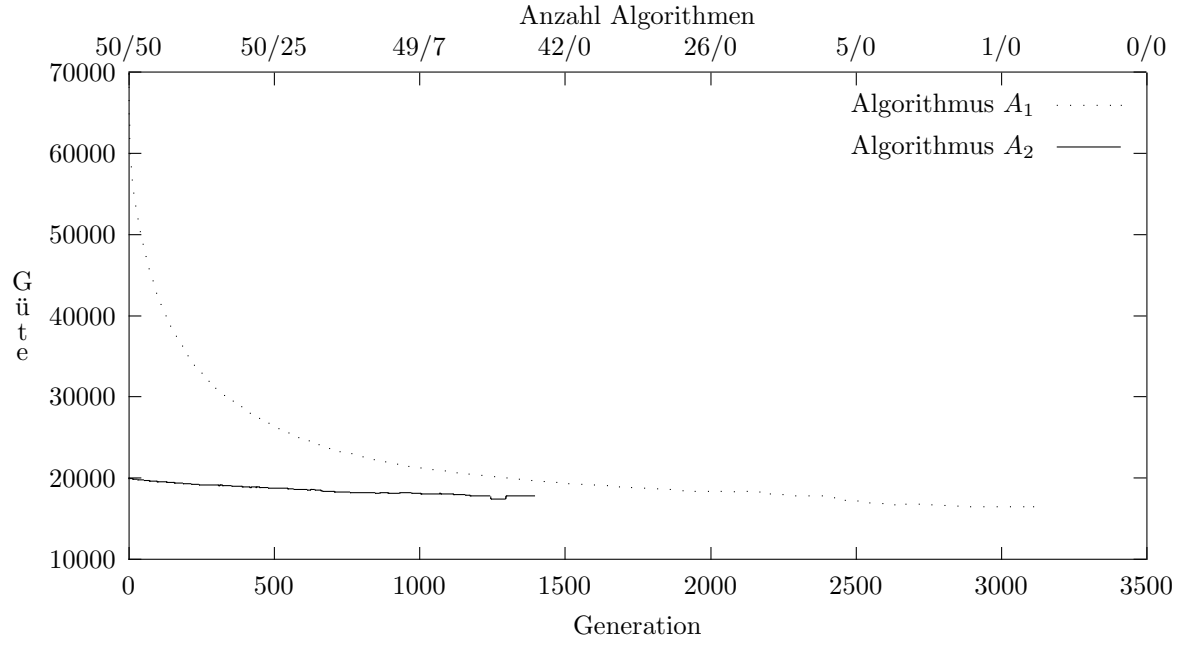


(a) Berechnungsverlauf für 50 Ausführungen von Algorithmus A_1 auf Produktionsdatensatz E. Der durchschnittliche Verlauf liegt etwas näher am Verlauf des Minimums als am Verlauf des Maximums.

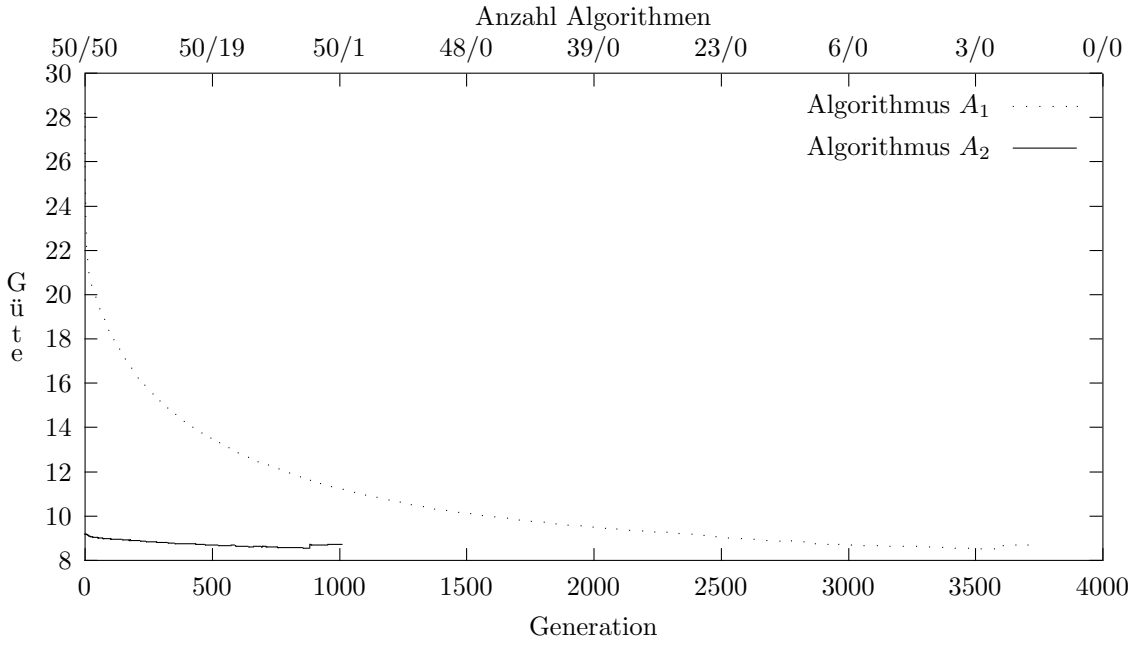


(b) Berechnungsverlauf für 50 Ausführungen von Algorithmus A_1 auf dem zufälligen Datensatz f. Der durchschnittliche Verlauf liegt bei niedrigerer Varianz mittig zwischen Minimum und Maximum.

Abbildung 5.1: Berechnungsverlauf für 50 Ausführungen von Algorithmus A_1 auf den Datensätzen E und f



(a) Berechnungsverlauf für jeweils 50 Ausführungen der Algorithmen A_1 und A_2 auf dem Produktionsdatensatz E. Bedingt durch die wesentlich besseren Startlösungen konvergiert Algorithmus A_2 deutlich langsamer als Algorithmus A_1, terminiert jedoch bei vergleichbarer Ergebnisgüte etwa doppelt so schnell wie dieser.



(b) Berechnungsverlauf für jeweils 50 Ausführungen der Algorithmen A_1 und A_2 auf dem zufälligen Datensatz f. Bedingt durch die wesentlich besseren Startlösungen konvergiert Algorithmus A_2 deutlich langsamer als Algorithmus A_1, terminiert jedoch bei vergleichbarer Ergebnisgüte in etwa einem Viertel der Zeit.

Abbildung 5.2: Berechnungsverlauf für 50 Ausführungen der Algorithmen A_1 und A_2 auf den Datensätzen E und f

DS	Ergebnisgüte (A_1 besser ⟵ ⟶ A_2 besser)
A	0,0408
B	0,0354
C	0,0275
D	0,0004
E	0,0162
F	0,0659
G	0,0772
H	0,0021
I	0,0070
J	0,0101
K	0,0014
Ø	0,0253

DS	Ergebnisgüte (A_1 besser ⟵ ⟶ A_2 besser)
a	0,0047
b	0,0205
c	0,0008
d	0,0351
e	0,0759
f	0,0730
g	0,0388
h	0,0323
i	0,0193
j	0,0984
k	0,0835
l	0,0510
m	0,0613
n	0,0132
o	0,1563
p	0,1251
q	0,0956
r	0,1810
Ø	0,0551

TABELLE 5.3: Direkter Vergleich der Ergebnisgüte von Algorithmus A_1 und Algorithmus A_2

Auffällig ist die durch die besseren Startlösungen bedingte wesentlich frühere Terminierung von Algorithmus A_2. Da das Verhalten der beiden Algorithmen auf zwei Datensätzen allein nur bedingt aussagekräftig ist, nehmen wir in Tabelle 5.3 einen direkten Vergleich der Ergebnisse von Algorithmus A_1 und Algorithmus A_2 vor.

Algorithmus A_1 zeigt auf den Produktionsdaten eine um 2,53% bessere Leistung als Algorithmus A_2, insbesondere auf den beiden ähnlichen (siehe Abschnitt 5.2) Datensätzen F und G. Auf den zufälligen Datensätzen liegt Algorithmus A_2 mit 5,51% vorne, besonders auf den größeren Datensätzen j bis r. Zusammenfassend lässt sich sagen, dass Algorithmus A_1 bei deutlich längerer Laufzeit auf strukturierten Daten geringfügig bessere Ergebnisse bringt, während Algorithmus A_2 auf zufälligen größeren Datensätzen deutlich schneller spürbar bessere Ergebnisse liefert. Tatsächlich liegen für Algorithmus A_1 auf Datensatz r aufgrund der langen Laufzeit weniger Ergebnisse vor als für die anderen Algorithmen (siehe Abschnitt 5.5.1).

Eine Ersetzung der linearen rangbasierten Selektion in Algorithmus A_2 durch die exponentielle rangbasierte Selektion mit $c = 0{,}9$ brachte deutlich schlechtere Ergebnisse auf den zufälligen Datensätzen und ein leicht schlechteres Verhalten auf den Produktionsdatensätzen. Das Weglassen der Identität als Mutationsoperator verschlechterte das Verhalten des Algorithmus ebenfalls.

5.4.2 Populationsbasierte Ansätze

In diesem Abschnitt stellen wir Ansätze vor, die auf dem Mutationsoperator 4.11 (populationsgesteuerte Inversion und Transposition) und damit auf dem inver-over Operator für das TSP basieren.

Wir bilden zuerst Algorithmus 4.10 mit den ursprünglichen Einstellungen aus dem Artikel von Tao und Michalewicz [54] nach. Wir setzen dazu die Populationsgröße auf $\mu = \lambda = 100$, die Wahrscheinlichkeit für eine zufällige Modifikation auf $w = 0{,}02$ und terminieren den Algorithmus

nach 10 Generationen ohne Verbesserung. Durch $\mu = \lambda$, die Verwendung der Komma-Selektion und der Beschränkung auf die Identität als einzigen Rekombinationsoperator erreichen wir eine mit jeder Generation wiederholte Anwendung von Mutationsoperator 4.11 auf eine gleich[5] bleibende Menge von Individuen. Da sowohl bei der Paarungsselektion als auch bei der Umweltselektion jeweils 100 aus 100 Individuen ausgewählt werden, ist die Wahl der Selektionsmethoden bedeutungslos. Als Startlösungen wählen wir zufällig erzeugte Individuen. Parametersatz 5.3 fasst die Einstellungen zusammen.

Parameter	A	R	Wert
Name			B_1 (populationsbasierter Ansatz, Variante 1)
Populationsgröße			$\mu = 100$
Anzahl der Kinder			$\lambda = 100$
Selektionsstrategie			(μ, λ)
Terminierung			Keine Verbesserung der besten Individuen ($\tau_{\text{last}} = 10$)
			Max. Anzahl an Auswertungen der Fitnessfunktion ($f_{\max} = 10000000$)
Initialisierung		1	Uniforme Verteilung
Paarungsselektion		1	Deterministische Selektion des besten Individuums
Umweltselektion		1	Deterministische Selektion des besten Individuums
Rekombination		1	Identität
Mutation		1	Populationsgesteuerte Inversion und Transposition ($w = 0{,}02$)

PARAMETERSATZ 5.3: Algorithmus B_1 (populationsbasierter Ansatz, Variante 1)

Der direkte Vergleich von Algorithmus B_1 mit Algorithmus A_2 in Tabelle 5.4 offenbart eine überaus schlechte Leistung von Algorithmus B_1: Bis auf wenige Ausnahmen, auch hier wieder die Datensätze F und G sowie der sehr kleine zufällige Datensatz a, schneidet Algorithmus B_1 durchweg wesentlich schlechter ab als Algorithmus A_2. Auf den Produktionsdaten beträgt die durchschnittliche Differenz 132,16%, auf den zufälligen Datensätzen sogar 287,13%! Dabei legen die Ergebnisse auf den zufälligen Datensätzen einen Zusammenhang zwischen Problemgröße und der Differenz der Ergebniswerte nahe.

Um weitere Hinweise auf die Ursachen für die schlechtere Leistung von Algorithmus B_1 zu erhalten, vergleichen wir in Abbildung 5.3 den Berechnungsverlauf beider Algorithmen auf dem Produktionsdatensatz E, auf dem Algorithmus B_1 etwa doppelt so schlecht abschneidet wie Algorithmus A_2. Der Kurvenverlauf legt nahe, dass die Terminierungsbedingung den Algorithmus zu früh beendet. In Parametersatz 5.4 setzen wir daher $\tau_{\text{last}} = 100$ und vergleichen den derart modifizierten Algorithmus in Tabelle 5.5 mit Algorithmus A_2.

Der Vergleich zeigt eine moderate Verbesserung der Leistung gegenüber Algorithmus B_1: Auf den Produktionsdaten schneidet Algorithmus B_2 um 25,57% schlechter ab als Algorithmus A_2, eine Verbesserung um 106,59% gegenüber Algorithmus B_1; bei den zufälligen Daten ist die Lage mit einer durchschnittlichen Differenz von 159,33% und einer Verbesserung um 127,8% ähnlich. Bei genauerer Betrachtung zeigt sich, dass Algorithmus B_2 auf neun von elf Produktionsdatensätzen bessere Ergebnisse bringt als Algorithmus A_2. Die schlechtere durchschnittliche Leistung geht auf das besonders schlechte Abschneiden auf den Datensätzen A und B zurück; hier scheint die Struktur der Probleminstanzen eine besondere Rolle zu spielen. Bei den zufälligen Datensätzen ist das Bild einheitlicher: Algorithmus B_2 zeigt, wie Algorithmus B_1 auch, eine mit zunehmender Problemgröße immer schlechter werdende Leistung.

[5] Die Individuen bleiben in dem Sinne gleich, dass sie von der Identität als Rekombinationsoperator unverändert übernommen werden. Durch die gewählten Parameter — Komma-Selektion, $\mu = \lambda$ und die Identität als einzigem Rekombinationsoperator — erreicht man eine Veränderung der Individuen allein durch die Mutationsoperatoren.

DS	Ergebnisgüte (A_2 besser ⟵ / ⟶ B_1 besser)
A	3.8420 ⟵
B	5.7840 ⟵
C	0.4858 ⟵
D	1.0573 ⟵
E	2.2050 ⟵
F	0.2020 ⟶
G	0.2523 ⟶
H	0.0208 ⟶
I	0.0154 ⟵
J	1.3813 ⟵
K	0.2420 ⟵
Ø	1.3216 ⟵

(a) Produktionsdaten

DS	Ergebnisgüte (A_2 besser ⟵ / ⟶ B_1 besser)
a	0.1205 ⟶
b	0.9497 ⟵
c	0.9683 ⟵
d	0.8797 ⟵
e	1.8720 ⟵
f	1.8747 ⟵
g	1.8786 ⟵
h	1.8240 ⟵
i	1.9308 ⟵
j	2.7887 ⟵
k	3.1727 ⟵
l	3.1144 ⟵
m	3.0346 ⟵
n	2.9361 ⟵
o	4.9563 ⟵
p	5.5646 ⟵
q	5.7956 ⟵
r	8.2629 ⟵
Ø	2.8713 ⟵

(b) Zufällige Datensätze

TABELLE 5.4: Direkter Vergleich der Ergebnisgüte von Algorithmus A_2 und Algorithmus B_1

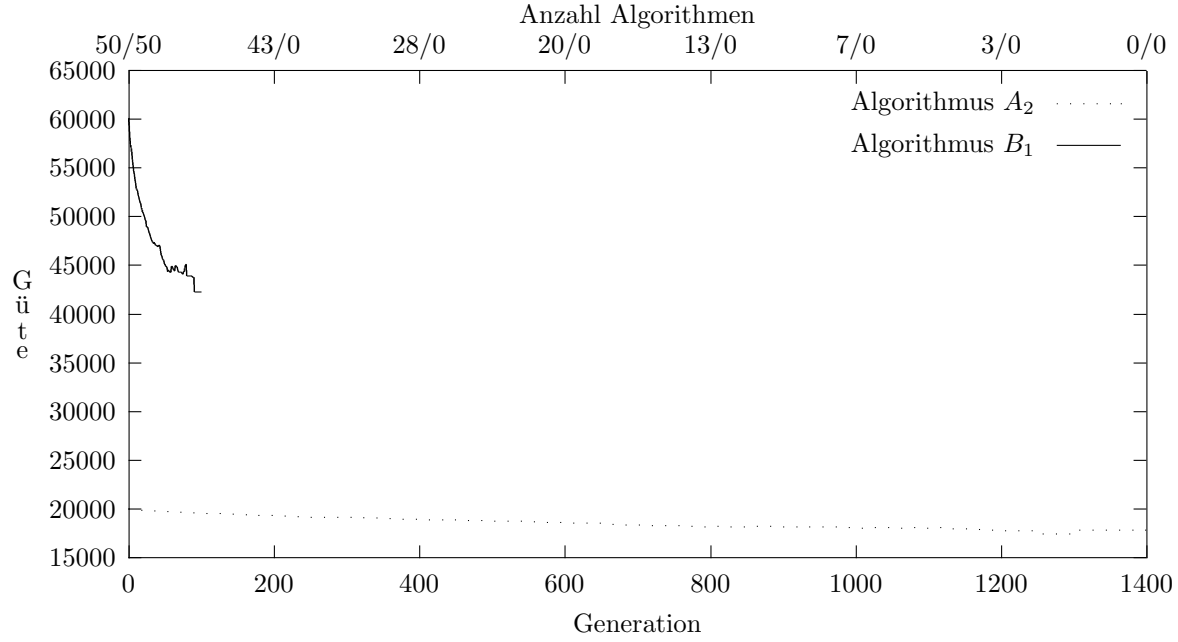


ABBILDUNG 5.3: Berechnungsverläufe für jeweils 50 Ausführungen der Algorithmen A_2 und B_1 auf dem Produktionsdatensatz E. Algorithmus B_1 konvergiert rasch, terminiert jedoch zu früh.

DS	Ergebnisgüte (A_2 besser ⟵ ⟶ B_2 besser)
A	1.1263 ⟵
B	3.1006 ⟵
C	0.1637 ⟶
D	0.2677 ⟶
E	0.0331 ⟶
F	0.2162 ⟶
G	0.2566 ⟶
H	0.0846 ⟶
I	0.0973 ⟶
J	0.0891 ⟶
K	0.2061 ⟶
Ø	0.2557 ⟶

(a) Produktionsdaten

DS	Ergebnisgüte (A_2 besser ⟵ ⟶ B_2 besser)
a	0.1721 ⟶
b	0.0663 ⟶
c	0.0466 ⟶
d	0.0112 ⟶
e	0.3970 ⟵
f	0.7990 ⟵
g	0.8633 ⟵
h	0.7104 ⟵
i	0.6745 ⟵
j	1.8837 ⟵
k	2.0466 ⟵
l	1.7735 ⟵
m	1.6569 ⟵
n	1.4381 ⟵
o	3.4619 ⟵
p	3.7453 ⟵
q	3.8051 ⟵
r	5.7209 ⟵
Ø	1.5933 ⟵

(b) Zufällige Datensätze

Tabelle 5.5: Direkter Vergleich der Ergebnisgüte von Algorithmus A_2 und Algorithmus B_2

Parameter	A	R	Wert
Name			B_2 (populationsbasierter Ansatz, Variante 2)
Populationsgröße			$\mu = 100$
Anzahl der Kinder			$\lambda = 100$
Selektionsstrategie			(μ, λ)
Terminierung			Keine Verbesserung der besten Individuen ($\tau_{\text{last}} = 100$)
			Max. Anzahl an Auswertungen der Fitnessfunktion ($f_{\max} = 10000000$)
Initialisierung		1	Uniforme Verteilung
Paarungsselektion		1	Deterministische Selektion des besten Individuums
Umweltselektion		1	Deterministische Selektion des besten Individuums
Rekombination		1	Identität
Mutation		1	Populationsgesteuerte Inversion und Transposition ($w = 0{,}02$)

Parametersatz 5.4: Algorithmus B_2 (populationsbasierter Ansatz, Variante 2). Gegenüber Algorithmus B_1 wurde τ_{last} auf 100 erhöht.

Als Anhaltspunkte zur weiteren Verbesserung des populationsbasierten Ansatzes verbleiben die Vergrößerung der Population, nahegelegt durch das Verhalten von Algorithmus A_2 auf den zufälligen Datensätzen, und die Verwendung besserer Startlösungen, nahegelegt durch das Vorgehen in Abschnitt 5.4.1. Die Vergrößerung der Population erbrachte keine wesentliche Verbesserung des Algorithmus. Wir ersetzen daher die zufällige Bestimmung der Startlösungen durch die nächster Roboter & gierige Ersparnis-Heuristik; Parametersatz 5.5 zeigt die entsprechenden Einstellungen.

Der direkte Vergleich des neuen Parametersatzes 5.5 mit Algorithmus A_2 in Tabelle 5.6 zeigt eine erneute Verbesserung gegenüber Algorithmus B_2: Auf den Produktionsdatensätzen A und B ist Algorithmus B_3 marginal schlechter als Algorithmus A_2, auf den anderen Produktionsdatensätzen dagegen zwischen 5,05% und 25,66% besser. Auf den zufälligen Datensätzen hat sich das Verhältnis bei weiterhin mit zunehmender Problemgröße schlechter werdender Leistung weiter zu Gunsten von Algorithmus B_3 verschoben: Dieser schlägt Algorithmus A_2 nun auf den Datensätzen a bis g sowie j und liegt im Schnitt etwa gleichauf mit Algorithmus A_2.

Eine Ursache für diese Ergebnislage liegt in den in Tabelle 5.7 aufgeführten Anzahlen der von den beiden Algorithmen vorgenommenen Funktionsauswertungen: Ein besseres Ergebnis auf einem Datensatz schlägt sich fast stets in einer deutlich höheren Anzahl von Auswertungen der Fitnessfunktion nieder. Auf den Produktionsdatensätzen C bis K liegt Algorithmus B_3 vorne und benötigt 10,44 bis 38,31 mal mehr Auswertungen der Fitnessfunktion als Algorithmus A_2. Bei den zufälligen Datensätzen verschiebt sich das Verhältnis der Funktionsauswertungen mit zunehmender Problemgröße, ähnlich wie bei den Ergebnisgüten; während bei Datensatz a Algorithmus B_3 mit der 51,75fachen Anzahl an Funktionsauswertungen ein um 16,99% besseres Ergebnis erzielt, ist es bei Datensatz r Algorithmus A_2 der mit dem 8,73fachen Aufwand ein um 8,23% besseres Ergebnis liefert.

DS	Ergebnisgüte (A_2 besser ← → B_3 besser)
A	0.0010
B	0.0269
C	0.1256
D	0.1365
E	0.1337
F	0.2162
G	0.2566
H	0.0837
I	0.0927
J	0.0505
K	0.1661
Ø	0.1121

(a) Produktionsdaten

DS	Ergebnisgüte (A_2 besser ← → B_3 besser)
a	0.1699
b	0.0944
c	0.0696
d	0.0435
e	0.0421
f	0.0128
g	0.0035
h	0.0472
i	0.0650
j	0.0019
k	0.0152
l	0.0366
m	0.0759
n	0.1606
o	0.0277
p	0.0486
q	0.1007
r	0.0823
Ø	0.0123

(b) Zufällige Datensätze

Tabelle 5.6: Direkter Vergleich der Ergebnisgüte von Algorithmus A_2 und Algorithmus B_3

Parameter	A	R	Wert
Name			B_3 (populationsbasierter Ansatz, Variante 3)
Populationsgröße			$\mu = 100$
Anzahl der Kinder			$\lambda = 100$
Selektionsstrategie			(μ, λ)
Terminierung			Keine Verbesserung der besten Individuen ($\tau_{\text{last}} = 100$)
			Max. Anzahl an Auswertungen der Fitnessfunktion ($f_{\max} = 10000000$)
Initialisierung		1	Nächster Roboter & gierige Ersparnis-Heuristik (probabilistisch)
Paarungsselektion		1	Deterministische Selektion des besten Individuums
Umweltselektion		1	Deterministische Selektion des besten Individuums
Rekombination		1	Identität
Mutation		1	Populationsgesteuerte Inversion und Transposition ($w = 0{,}02$)

PARAMETERSATZ 5.5: Algorithmus B_3 (populationsbasierter Ansatz, Variante 3). Gegenüber Algorithmus B_2 wurden die zufälligen Startlösungen durch heuristisch bestimmte ersetzt.

Zusammengefasst liefert Algorithmus B_3 auf kleineren, Algorithmus A_2 auf größeren Datensätzen bessere Ergebnisse, wobei die Verbesserung in der Ergebnisgüte stets mit einer erheblich größeren Anzahl an Auswertungen der Fitnessfunktion erkauft wird.

	Funktionsauswertungen			
DS	Alg. A_2	Alg. B_3	Quotienten	
A	14161	37555	0,38	2,65
B	26866	28836	0,93	1,07
C	1422	53351	0,03	37,51
D	3033	50549	0,06	16,66
E	4078	50617	0,08	12,41
F	661	12220	0,05	18,47
G	525	12265	0,04	23,33
H	934	34278	0,03	36,69
I	975	37362	0,03	38,31
J	3559	37167	0,10	10,44
K	1287	35191	0,04	27,34
Ø	5227	35399	0,15	6,77

(a) Produktionsdaten

	Funktionsauswertungen			
DS	Alg. A_2	Alg. B_3	Quotienten	
a	393	20381	0,02	51,75
b	3053	35640	0,09	11,67
c	2464	24219	0,10	9,83
d	2779	27431	0,10	9,87
e	4013	27568	0,15	6,87
f	5305	20247	0,26	3,82
g	6051	18988	0,32	3,14
h	7517	18871	0,40	2,51
i	8173	19097	0,43	2,34
j	12728	20226	0,63	1,59
k	16496	20343	0,81	1,23
l	16756	18129	0,92	1,08
m	20764	18322	1,13	0,88
n	24793	17614	1,41	0,71
o	52448	17346	3,02	0,33
p	61742	22051	2,80	0,36
q	76176	17832	4,27	0,23
r	175064	20062	8,73	0,11
Ø	27595	21354	1,29	0,77

(b) Zufällige Datensätze

TABELLE 5.7: Anzahl der Auswertungen der Fitnessfunktion für Algorithmus A_2 und Algorithmus B_3

5.4.3 Simuliertes Abkühlen

Wir bilden das simulierte Abkühlen mit Hilfe des in Kapitel 4 entworfenen Mutationsoperators (Algorithmus 4.12) wie folgt in unserem Ansatz ab: Da beim simulierten Abkühlen nur mit einer einzigen, iterativ variierten Lösung gearbeitet wird, setzen wir die Populationsgröße auf ein einzelnes Individuum, ebenso die Anzahl der Kinder. Das Elternindividuum entspricht der Lösung vor der Modifikation, das Kindindividuum der Lösung nach der Modifikation. Da die Modifikation der Lösung allein durch den Mutationsoperator erfolgen soll, verwenden wir wie in Abschnitt 5.4.2 ausschließlich die Identität als Rekombinationsoperator und stellen das Elternindividuum dem Mutationsoperator so unverändert zur Verfügung. Die Selektionsmethoden sind bedeutungslos, da nur ein Individuum zur Verfügung steht; wir verwenden sowohl zur Paarungs- als auch zur Umweltselektion die deterministische Selektion des besten Individuums. Als Terminierungsbedingung wählen wir wie in Abschnitt 5.4.1 die Terminierung nach 100 Generationen ohne Verbesserung, als Startlösung ein zufällig erzeugtes Individuum. Den Parameter für den exponentiellen Abkühlungsverlauf setzen wir auf $\alpha = 0{,}9$.

Es verbleibt die Wahl der über den Parameter w kontrollierten zulässigen lokalen Züge. Wir definieren zu diesem Zweck drei Parametersätze mit $w = 1$ (nur Transpositionen), $w = 0$ (nur Inversionen) und $w = 1/2$ (Transpositionen und Inversionen mit gleicher Wahrscheinlichkeit). Es ergeben sich die drei Parametersätze 5.6, 5.7 und 5.8.

Parameter	A	R	Wert
Name			C_1 (simuliertes Abkühlen, Variante 1)
Populationsgröße			$\mu = 1$
Anzahl der Kinder			$\lambda = 1$
Selektionsstrategie			(μ, λ)
Terminierung			Keine Verbesserung der besten Individuen ($\tau_{\text{last}} = 100$)
			Max. Anzahl an Auswertungen der Fitnessfunktion ($f_{\max} = 10000000$)
Initialisierung		1	Uniforme Verteilung
Paarungsselektion		1	Deterministische Selektion des besten Individuums
Umweltselektion		1	Deterministische Selektion des besten Individuums
Rekombination		1	Identität
Mutation		1	Simuliertes Abkühlen ($\alpha = 0{,}9, w = 1$)

Parametersatz 5.6: Algorithmus C_1 (simuliertes Abkühlen, Variante 1)

Parameter	A	R	Wert
Name			C_2 (simuliertes Abkühlen, Variante 2)
Populationsgröße			$\mu = 1$
Anzahl der Kinder			$\lambda = 1$
Selektionsstrategie			(μ, λ)
Terminierung			Keine Verbesserung der besten Individuen ($\tau_{\text{last}} = 100$)
			Max. Anzahl an Auswertungen der Fitnessfunktion ($f_{\max} = 10000000$)
Initialisierung		1	Uniforme Verteilung
Paarungsselektion		1	Deterministische Selektion des besten Individuums
Umweltselektion		1	Deterministische Selektion des besten Individuums
Rekombination		1	Identität
Mutation		1	Simuliertes Abkühlen ($\alpha = 0{,}9, w = 0$)

Parametersatz 5.7: Algorithmus C_2 (simuliertes Abkühlen, Variante 2)

DS	Ergebnisgüte (C_1 besser ← → C_3 besser)	
A		0.1506
B		0.3797
C		0.0544
D		0.0977
E		0.3320
F	0.0555	
G	0.0801	
H	0.0166	
I	0.0228	
J		0.1826
K		0.0121
Ø		0.0940

(a) Produktionsdaten

DS	Ergebnisgüte (C_1 besser ← → C_3 besser)	
a		0.0146
b		0.1382
c		0.1199
d		0.0810
e		0.2067
f		0.2291
g		0.1442
h		0.0467
i	0.0640	
j		0.3275
k		0.3341
l		0.0842
m	0.0687	
n	0.9440	
o		0.1514
p		0.1508
q	2.4340 ←	
r	4.3273 ←	
Ø	0.3228	

(b) Zufällige Datensätze

TABELLE 5.8: Direkter Vergleich der Ergebnisgüte von Algorithmus C_1 und Algorithmus C_3

Parameter	A	R	Wert
Name			C_3 (simuliertes Abkühlen, Variante 3)
Populationsgröße			$\mu = 1$
Anzahl der Kinder			$\lambda = 1$
Selektionsstrategie			(μ, λ)
Terminierung			Keine Verbesserung der besten Individuen ($\tau_{\text{last}} = 100$)
			Max. Anzahl an Auswertungen der Fitnessfunktion ($f_{\max} = 10000000$)
Initialisierung		1	Uniforme Verteilung
Paarungsselektion		1	Deterministische Selektion des besten Individuums
Umweltselektion		1	Deterministische Selektion des besten Individuums
Rekombination		1	Identität
Mutation		1	Simuliertes Abkühlen ($\alpha = 0{,}9, w = 0{,}5$)

PARAMETERSATZ 5.8: Algorithmus C_3 (simuliertes Abkühlen, Variante 3)

Abbildung 5.4 zeigt die Berechnungsverläufe der drei Varianten des simulierten Abkühlens auf dem Produktionsdatensatz A und dem zufälligen Datensatz f. Auffällig ist die langsamere Konvergenz von Algorithmus C_2. Wir führen dies darauf zurück, dass Knoten durch Inversionen allein nicht von einer Partition zur anderen verschoben werden können und der Algorithmus dadurch nur die Knoten innerhalb der einzelnen Partitionen optimieren kann. Dieser Umstand wird in der Abbildung durch die Durchschnittsbildung über mehrere Berechnungsverläufe etwas abgemildert.

Für die Algorithmen C_1 und C_3 führen wir in Tabelle 5.8 einen direkten Vergleich der Ergebnisse durch. Auf den Produktionsdaten zeigt sich ein leichter Vorsprung von Algorithmus C_3 um durchschnittlich 9,4%; auf den zufälligen Daten ist die Lage weniger eindeutig: Zwar liefert Algorithmus C_1 durchschnittlich um 32,28% bessere Ergebnisse, jedoch wird dieser Durchschnitt im Wesentlichen durch die Datensätze n, q und r bestimmt; auf der Mehrzahl der Datensätze ist Algorithmus C_3 im Vorteil.

Wir entscheiden uns aufgrund der besseren Leistung auf den Produktionsdatensätzen für Algorithmus C_3. Abbildung 5.4 lässt vermuten, dass sich die Ergebnisse des Algorithmus verbessern lassen, wenn man die Anzahl der simulierten Generationen erhöht. Wir setzen dazu $\tau_{\text{last}} = 800$ (Parametersatz 5.9) und führen in Tabelle 5.9 einen direkten Vergleich mit Algorithmus C_3 durch.

Die Güte der Ergebnisse erhöht sich durch die längere Laufzeit beträchtlich (um 47,24% auf den Produktionsdaten, um 168,36% auf den zufälligen Daten), ebenso die in Tabelle 5.10 aufgeführte Anzahl der Funktionsauswertungen: Algorithmus C_4 benötigt auf den Produktionsdaten im Schnitt 6,77 mal so viele Funktionsauswertungen wie Algorithmus C_3, auf den zufälligen Datensätzen gar 12,2 mal so viele.

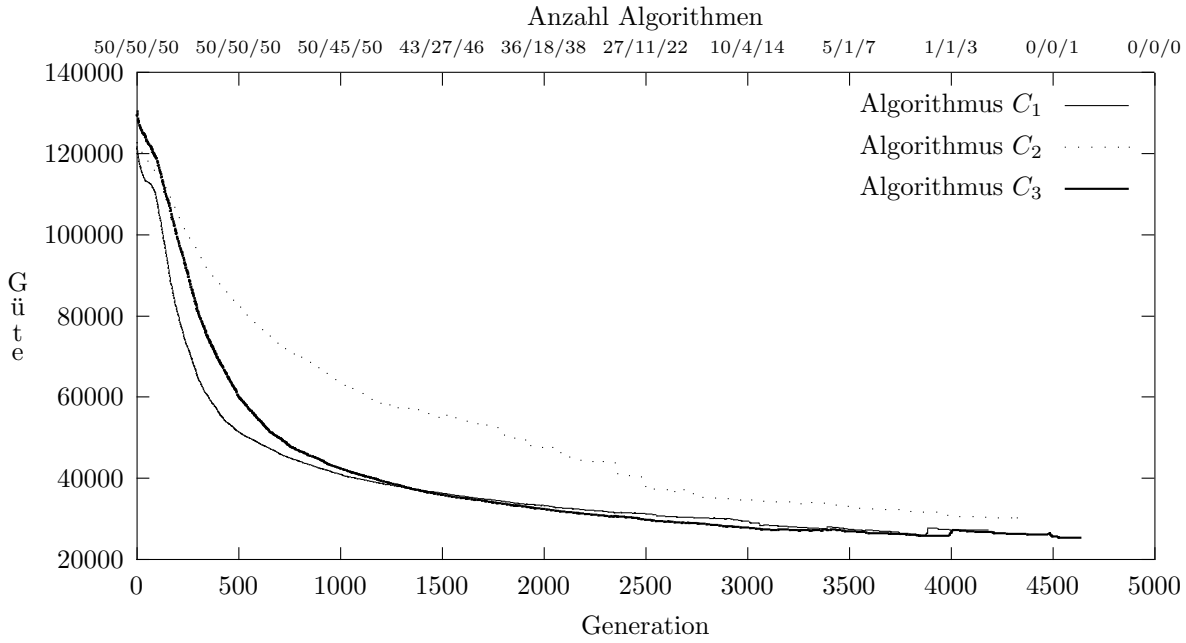
DS	Ergebnisgüte (C_3 besser ← → C_4 besser)	
A	→	1.2064
B	→	2.0181
C		0.1277
D		0.2760
E		0.6660
F		0.1752
G		0.1794
H		0.0479
I		0.0721
J		0.3043
K		0.1232
Ø		0.4724

(a) Produktionsdaten

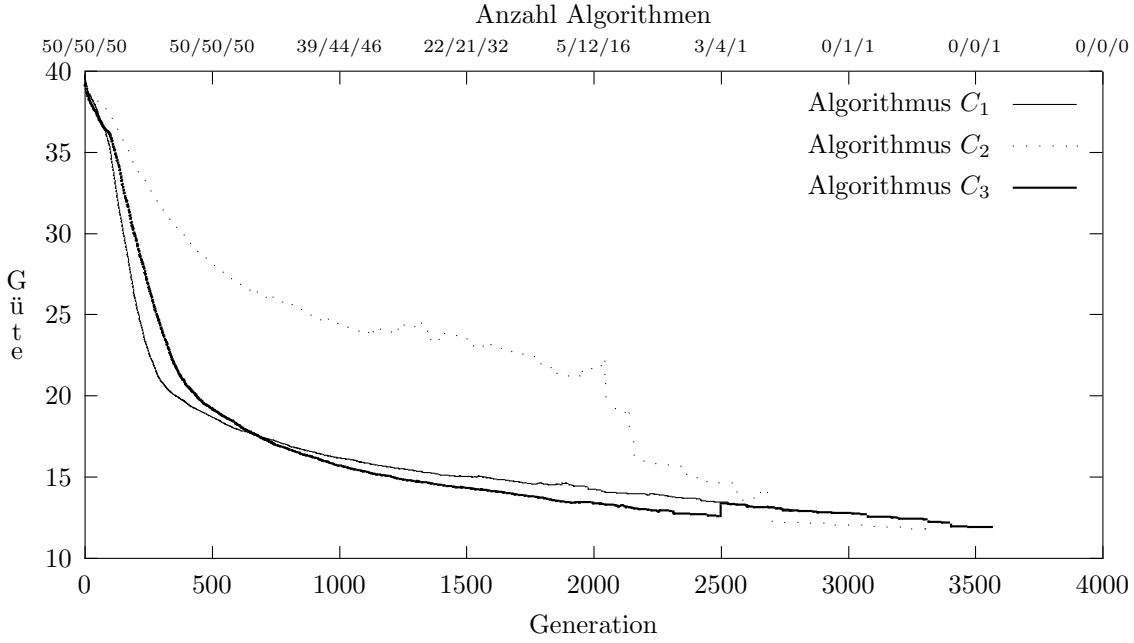
DS	Ergebnisgüte (C_3 besser ← → C_4 besser)	
a		0.0108
b		0.3469
c		0.4301
d		0.4677
e		0.6676
f		0.8086
g		0.8987
h		0.9731
i	→	1.0449
j	→	1.1486
k	→	1.3585
l	→	1.4223
m	→	1.4597
n	→	2.1801
o	→	2.0934
p	→	2.1454
q	→	4.7123
r	→	8.1370
Ø	→	1.6836

(b) Zufällige Datensätze

Tabelle 5.9: Direkter Vergleich der Ergebnisgüte von Algorithmus C_3 und Algorithmus C_4



(a) Berechnungsverläufe für jeweils 50 Ausführungen der Algorithmen C_1, C_2 und C_3 auf dem Produktionsdatensatz A. Algorithmus C_2 konvergiert langsamer als die beiden anderen Algorithmen.



(b) Berechnungsverläufe für jeweils 50 Ausführungen der Algorithmen C_1, C_2 und C_3 auf dem zufälligen Datensatz f. Algorithmus C_2 konvergiert deutlich langsamer als die beiden anderen Algorithmen.

Abbildung 5.4: Berechnungsverläufe für jeweils 50 Ausführungen der Algorithmen C_1, C_2 und C_3 auf den Datensätzen A und f

DS	Funktionsauswertungen			
	Alg. C_3	Alg. C_4	Quotienten	
A	2472	19693	0,13	7,96
B	3790	37289	0,10	9,84
C	1269	5532	0,23	4,36
D	1395	7889	0,18	5,65
E	1838	10640	0,17	5,79
F	324	1234	0,26	3,80
G	298	1178	0,25	3,94
H	357	1717	0,21	4,81
I	440	2153	0,20	4,89
J	1753	8700	0,20	4,96
K	681	2921	0,23	4,29
Ø	1329	8995	0,15	6,77

(a) Produktionsdaten

DS	Funktionsauswertungen			
	Alg. C_3	Alg. C_4	Quotienten	
a	294	1127	0,26	3,83
b	1127	6077	0,19	5,39
c	1045	6716	0,16	6,42
d	997	6459	0,15	6,47
e	1939	13488	0,14	6,96
f	1668	14493	0,12	8,69
g	1548	14633	0,11	9,45
h	1394	13289	0,10	9,53
i	1340	12375	0,11	9,23
j	3041	26588	0,11	8,74
k	2731	28514	0,10	10,44
l	2235	26640	0,08	11,92
m	2129	24405	0,09	11,46
n	1629	21410	0,08	13,14
o	4326	60383	0,07	13,96
p	4423	57046	0,08	12,90
q	2690	49989	0,05	18,58
r	3759	83748	0,04	22,27
Ø	2129	25965	0,08	12,20

(b) Zufällige Datensätze

Tabelle 5.10: Anzahl der Auswertungen der Fitnessfunktion für Algorithmus C_3 und Algorithmus C_4

Parameter	A	R	Wert
Name			C_4 (simuliertes Abkühlen, Variante 4)
Populationsgröße			$\mu = 1$
Anzahl der Kinder			$\lambda = 1$
Selektionsstrategie			(μ, λ)
Terminierung			Keine Verbesserung der besten Individuen ($\tau_{\text{last}} = 800$)
			Max. Anzahl an Auswertungen der Fitnessfunktion ($f_{\max} = 10000000$)
Initialisierung		1	Uniforme Verteilung
Paarungsselektion		1	Deterministische Selektion des besten Individuums
Umweltselektion		1	Deterministische Selektion des besten Individuums
Rekombination		1	Identität
Mutation		1	Simuliertes Abkühlen ($\alpha = 0{,}9, w = 0{,}5$)

Parametersatz 5.9: Algorithmus C_4 (simuliertes Abkühlen, Variante 4). Gegenüber Algorithmus C_3 wurde τ_{last} auf 800 erhöht.

DS	Ergebnisgüte (C_4 besser ⟵ ⟶ C_5 besser)	
A		0.4961
B		0.5573
C		0.0745
D		0.3181
E		0.0819
F		
G		0.0137
H		0.0056
I		0.0032
J		0.0187
K	0.0068	
Ø		0.1420

(a) Produktionsdaten

DS	Ergebnisgüte (C_4 besser ⟵ ⟶ C_5 besser)	
a		0.0613
b		0.0377
c		0.1106
d		0.0515
e		0.1310
f		0.0648
g		0.0396
h		0.1231
i		0.1570
j		0.1116
k		0.1512
l		0.3718
m		0.0978
n		0.4999
o		0.7112
p		0.2782
q		0.9848
r	⟶	1.3482
Ø		0.2962

(b) Zufällige Datensätze

TABELLE 5.11: Direkter Vergleich der Ergebnisgüte von Algorithmus C_4 und Algorithmus C_5

Parameter	A	R	Wert
Name			C_5 (simuliertes Abkühlen, Variante 5)
Populationsgröße			$\mu = 1$
Anzahl der Kinder			$\lambda = 1$
Selektionsstrategie			(μ, λ)
Terminierung			Keine Verbesserung der besten Individuen ($\tau_{\text{last}} = 800$)
			Max. Anzahl an Auswertungen der Fitnessfunktion ($f_{\max} = 10000000$)
Initialisierung		1	Nächster Roboter & gierige Ersparnis-Heuristik (deterministisch)
Paarungsselektion		1	Deterministische Selektion des besten Individuums
Umweltselektion		1	Deterministische Selektion des besten Individuums
Rekombination		1	Identität
Mutation		1	Simuliertes Abkühlen ($\alpha = 0{,}9, w = 0{,}5$)

PARAMETERSATZ 5.10: Algorithmus C_5 (simuliertes Abkühlen, Variante 5). Gegenüber Algorithmus C_4 wurde die zufällige Startlösung durch eine heuristisch bestimmte ersetzt.

DS	Funktionsauswertungen			
	Alg. C_4	Alg. C_5	Quotienten	
A	19693	19096	1,03	0,97
B	37289	32646	1,14	0,88
C	5532	5682	0,97	1,03
D	7889	7463	1,06	0,95
E	10640	10953	0,97	1,03
F	1234	1248	0,99	1,01
G	1178	1231	0,96	1,04
H	1717	1605	1,07	0,93
I	2153	2150	1,00	1,00
J	8700	8464	1,03	0,97
K	2921	2664	1,10	0,91
Ø	8995	8473	1,06	0,94

(a) Produktionsdaten

DS	Funktionsauswertungen			
	Alg. C_4	Alg. C_5	Quotienten	
a	1127	1129	1,00	1,00
b	6077	5995	1,01	0,99
c	6716	6414	1,05	0,96
d	6459	6396	1,01	0,99
e	13488	13977	0,97	1,04
f	14493	14331	1,01	0,99
g	14633	13738	1,07	0,94
h	13289	12561	1,06	0,95
i	12375	12458	0,99	1,01
j	26588	25298	1,05	0,95
k	28514	25797	1,11	0,90
l	26640	21673	1,23	0,81
m	24405	22107	1,10	0,91
n	21492	19002	1,13	0,88
o	60383	44089	1,37	0,73
p	57046	37271	1,53	0,65
q	49989	27127	1,84	0,54
r	83748	12106	6,92	0,14
Ø	25970	17859	1,45	0,69

(b) Zufällige Datensätze

Tabelle 5.12: Anzahl der Auswertungen der Fitnessfunktion für Algorithmus C_4 und Algorithmus C_5

Zuletzt ersetzen wir die zufällige Startlösung durch eine gemäß der nächster Roboter & gierige Ersparnis-Heuristik ermittelte Lösung. Parametersatz 5.10 fasst die Einstellungen zusammen; Tabelle 5.11 gibt einen direkten Vergleich von Algorithmus C_4 mit Algorithmus C_5. Die bessere Startlösung sorgt für eine weitere Verbesserung der Ergebnisse (14,2% auf den Produktionsdaten und 29,62% auf den zufälligen Daten); Tabelle 5.12 zeigt, dass dabei gleichzeitig Funktionsauswertungen eingespart wurden (6% weniger Funktionsauswertungen gegenüber Algorithmus C_4 auf den Produktionsdaten, 31% weniger Auswertungen auf den zufälligen Daten).

Von den fünf vorgestellten Varianten des simulierten Abkühlens hat sich mit Algorithmus C_5 die Variante mit einer guten Startlösung, Inversionen und Transpositionen als zulässigen lokalen Zügen und einem hohen τ_{last}-Wert als die erfolgreichste erwiesen. Eine Veränderung der Gewichtung von Inversion und Transposition ($w \in \{0{,}25, 0{,}75\}$) bewirkte keine Veränderung im Verhalten des Algorithmus. Eine Erhöhung des Parameters zur Kontrolle des Abkühlungsverlaufs auf $\alpha = 0{,}99$ verlängerte die Laufzeit des Algorithmus ohne die Ergebnisse zu verbessern. Einen Vergleich von Algorithmus C_5 mit den Varianten der anderen Verfahren nehmen wir im Rahmen der Gesamtauswertung (siehe Abschnitt 5.5) vor.

5.4.4 Iterierte lokale Suche

Wir geben sowohl eine stochastische Variante der iterierten lokalen Suche, die sich allein mit den Mutationsoperatoren Inversion und Transposition umsetzen lässt, als auch einen Parametersatz für die iterierte lokale Suche mit Hilfe eines maßgeschneiderten Mutationsoperators.

5.4.4.1 Eine stochastische Variante

Wir verzichten auf die Überprüfung aller Kanten (Nachbarn) eines Individuums und beschränken uns stattdessen auf Stichproben unter diesen. Wie in den Abschnitten 5.4.2 und 5.4.3 erreichen wir die Modifikation des einzigen Individuums allein durch die Mutationsoperatoren, indem wir $\mu = \lambda = 1$ setzen und als einzigen Rekombinationsoperator die Identität zulassen. Im Gegensatz zu den vorherigen Abschnitten verwenden wir die Plus-Selektion; dadurch werden Veränderungen des Individuums nur übernommen, wenn sie seine Fitness verbessern, was dem in Abschnitt 4.4.10.6 beschriebenen diskreten Gradientenabstiegsverfahren entspricht.

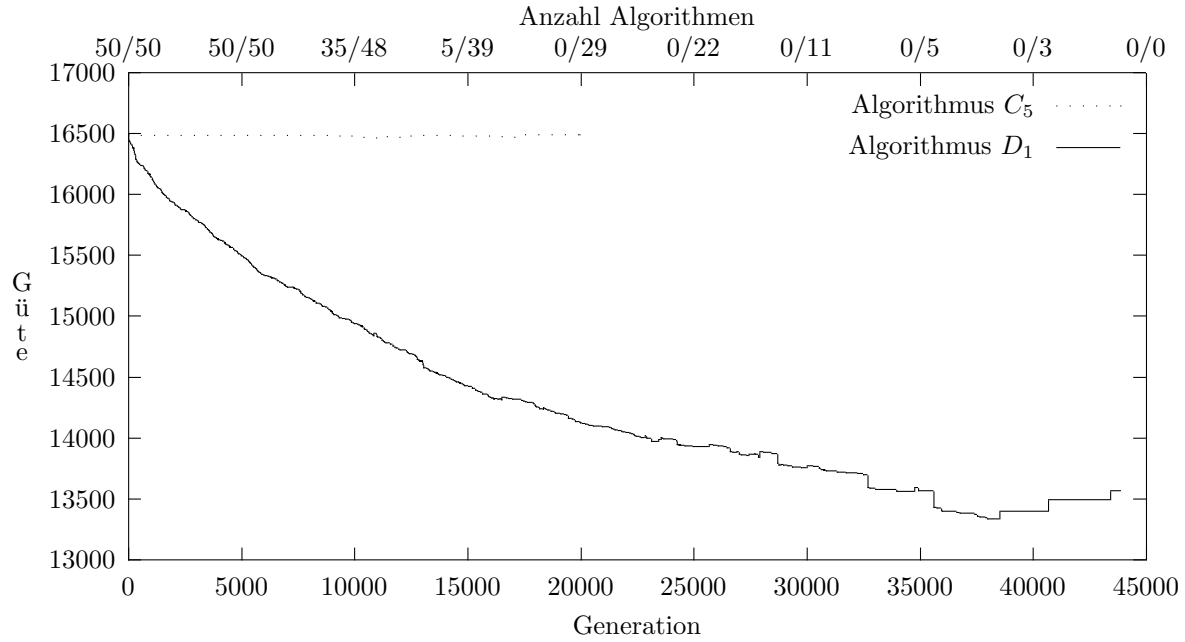
Anders als bei Algorithmus 4.13 können wir in dieser Variante nicht zwischen der (auf Transpositionen beschränkten) lokalen Suche und den gelegentlichen größeren Modifikationen der Lösung (bei denen, kontrolliert durch die Parameter s und w, sowohl Transpositionen als auch Inversionen zum Einsatz kommen) unterscheiden. Wir stellen als Mutationsoperatoren zu gleichen Teilen Inversion und Transposition zur Verfügung, da sich dies bereits beim simulierten Abkühlen bewährt hat (siehe Abschnitt 5.4.3).

Als Terminierungsbedingung verwenden wir das Kriterium „Keine Verbesserung der besten Individuen“. Dabei ist τ_{last} die maximale Anzahl durchgeführter Stichproben mit Wiederholung innerhalb der Nachbarschaft eines Individuums. Man beachte, dass es sich durch die Hinzunahme der Inversion um einen dichteren Nachbarschaftsgraphen als bei der Verwendung der Transposition allein handelt. Da in den vorherigen Abschnitten die Ersetzung zufälliger Startlösungen durch von der nächster Roboter & gierige Ersparnis-Heuristik bestimmte Individuen stets zu einer Verbesserung der Ergebnisse geführt hat, entscheiden wir uns bei der Bestimmung der Startlösung für diese. Parametersatz 5.11 fasst die Einstellungen zusammen.

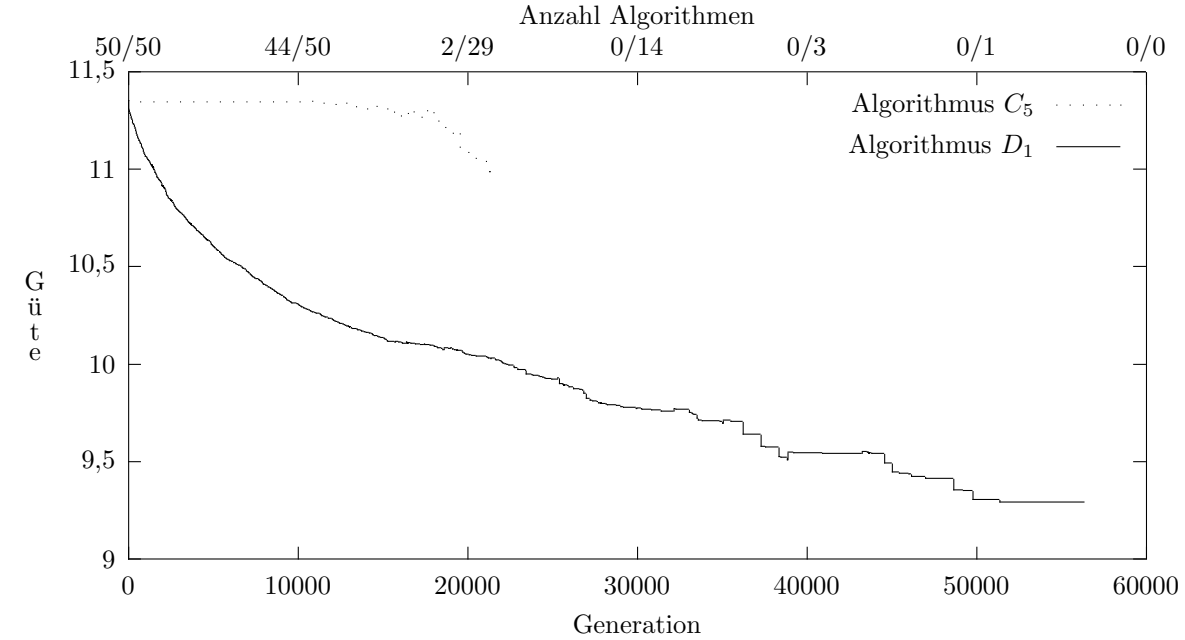
Bedingt durch den hohen Wert für τ_{last} wird Algorithmus D_1 wesentlich mehr Generationen simulieren als die anderen bisher betrachteten Algorithmen. Wir vergleichen den Berechnungsverlauf von Algorithmus D_1 daher in Abbildung 5.5 mit dem von Algorithmus C_5, der im Vergleich ebenfalls eine höhere Anzahl simulierter Generationen aufweist (siehe Abbildungen 5.1, 5.2, 5.3 und 5.4). Wie erwartet führt Algorithmus D_1 deutlich mehr Simulationsschritte aus (auf den betrachteten Datensätzen grob doppelt so viele wie Algorithmus C_5), erbringt gleichzeitig aber auch wesentlich bessere Ergebnisse. Tabelle 5.13 zeigt einen direkten Vergleich von Algorithmus D_1 mit Algorithmus C_5. Auf den Produktionsdaten liegt Algorithmus D_1 um 5,89% vor Algorithmus C_5, auf den zufälligen Daten um 30,6%, wobei bei den größeren Datensätzen auch prozentuell größere Verbesserungen möglich waren. Man beachte den Gleichstand bei Datensatz F, ein Hinweis auf ein mögliches globales Optimum (siehe auch Tabelle 5.17).

Parameter	A	R	Wert
Name			D_1 (iterierte lokale Suche, Variante 1)
Populationsgröße			$\mu = 1$
Anzahl der Kinder			$\lambda = 1$
Selektionsstrategie			$(\mu + \lambda)$
Terminierung			Keine Verbesserung der besten Individuen ($\tau_{\text{last}} = 5000$)
			Max. Anzahl an Auswertungen der Fitnessfunktion ($f_{\max} = 10000000$)
Initialisierung		1	Nächster Roboter & gierige Ersparnis-Heuristik (deterministisch)
Paarungsselektion		1	Deterministische Selektion des besten Individuums
Umweltselektion		1	Deterministische Selektion des besten Individuums
Rekombination		1	Identität
Mutation		1	Inversion
		1	Transposition

Parametersatz 5.11: Algorithmus D_1 (iterierte lokale Suche, Variante 1)



(a) Berechnungsverläufe für jeweils 50 Ausführungen der Algorithmen C_5 und D_1 auf dem Produktionsdatensatz E. Algorithmus D_1 liefert bei deutlich längerem Verlauf der Berechnung ein wesentlich besseres Ergebnis. Zum Anstieg der Verlaufskurve für Algorithmus D_1 gegen Ende der Berechnung siehe Abschnitt 5.3.4.



(b) Berechnungsverläufe für jeweils 50 Ausführungen der Algorithmen C_5 und D_1 auf dem zufälligen Datensatz e. Algorithmus D_1 liefert bei deutlich längerem Verlauf der Berechnung ein wesentlich besseres Ergebnis.

Abbildung 5.5: Berechnungsverläufe für jeweils 50 Ausführungen der Algorithmen C_5 und D_1 auf den Datensätzen E und e

DS	Ergebnisgüte (C_5 besser ⟵ ⟶ D_1 besser)	
A		0,0931
B		0,0955
C		0,0324
D		0,0001
E		0,1681
F	0,0000	0,0000
G		0,0049
H		0,0299
I		0,0402
J		0,0927
K		0,0907
Ø		0,0589

(a) Produktionsdaten

DS	Ergebnisgüte (C_5 besser ⟵ ⟶ D_1 besser)	
a		0,0847
b		0,1285
c		0,1574
d		0,2081
e		0,1388
f		0,2513
g		0,2689
h		0,3091
i		0,3277
j		0,3464
k		0,4252
l		0,3192
m		0,5777
n		0,2979
o		0,2321
p		0,8508
q		0,4187
r		0,1662
Ø		0,3060

(b) Zufällige Datensätze

Tabelle 5.13: Direkter Vergleich der Ergebnisgüte von Algorithmus C_5 und Algorithmus D_1

Parameter	A	R	Wert
Name			D_2 (iterierte lokale Suche, Variante 2)
Populationsgröße			$\mu = 1$
Anzahl der Kinder			$\lambda = 1$
Selektionsstrategie			(μ, λ)
Terminierung			Keine Verbesserung der besten Individuen ($\tau_{\text{last}} = 5$)
			Max. Anzahl an Auswertungen der Fitnessfunktion ($f_{\max} = 10000000$)
Initialisierung	1		Nächster Roboter & gierige Ersparnis-Heuristik (deterministisch)
Paarungsselektion	1		Deterministische Selektion des besten Individuums
Umweltselektion	1		Deterministische Selektion des besten Individuums
Rekombination	1		Identität
Mutation	1		Iterierte lokale Suche ($s = 5, w = 0{,}5$)

Parametersatz 5.12: Algorithmus D_2 (iterierte lokale Suche, Variante 2)

5.4.4.2 Iterierte lokale Optimierung

Wir verwenden den in Abschnitt 4.4.10.6 vorgestellten Mutationsoperator für die iterierte lokale Suche. Man beachte, dass dieser Operator im Gegensatz zum Vorgehen in Abschnitt 5.4.4.1 keine Inversionen als lokale Züge zulässt. Wir behalten die Einstellungen aus Parametersatz 5.11 bis auf τ_{last} bei und ersetzen lediglich die Mutationsoperatoren durch Algorithmus 4.13. Bei der Wahl von s und w entscheiden wir uns für $s = 5$ aufeinanderfolgende Mutationen zur Veränderung des Individuums zwischen den Gradientenabstiegen, wobei wir mit $w = 0{,}5$ Inversionen und Mutationen gleichermaßen zulassen. Da eine Durchführung von Algorithmus 4.13, d. h. die Simulation einer Generation, der kompletten Durchführung eines lokalen Optimierungsverfahrens entspricht, setzen wir τ_{last} auf einen sehr niedrigen Wert. Parametersatz 5.12 fasst die Einstellungen zusammen.

Ein Vergleich der Berechnungsverläufe von Algorithmus D_2 und Algorithmus D_1 wäre nicht sinnvoll, da Algorithmus D_1 sehr viele, Algorithmus D_2 dagegen sehr wenige Generationen simuliert. Wir führen daher in Tabelle 5.14 einen direkten Vergleich der Ergebnisse der beiden Algorithmen durch. Dieser zeigt eine leichte aber deutliche Verbesserung um 1,74% von Algorithmus D_2 auf den Produktionsdaten und eine deutliche Verbesserung um 17,1% auf den zufälligen Datensätzen. Auf den drei Produktionsdatensätzen D, F und G erreichen beide Algorithmen die gleiche Ergebnisgüte, ein weiterer Hinweis auf globale Optima (siehe auch Tabelle 5.17).

DS	Ergebnisgüte: D_1 besser ←	→ D_2 besser
A		0,0705
B		0,0879
C		0,0019
D	0,0000	0,0000
E	0,0415	
F	0,0000	0,0000
G	0,0000	0,0000
H		0,0040
I	0,0037	
J		0,0252
K		0,0474
Ø		0,0174

(a) Produktionsdaten

DS	Ergebnisgüte: D_1 besser ←	→ D_2 besser
a		0,0005
b	0,0062	
c	0,0316	
d		0,0333
e	0,0155	
f		0,0088
g		0,0444
h		0,0767
i		0,0610
j		0,1171
k		0,0973
l		0,0735
m		0,1461
n		0,1003
o		0,3579
p		0,4177
q		0,4379
r		1,1592
Ø		0,1710

(b) Zufällige Datensätze

Tabelle 5.14: Direkter Vergleich der Ergebnisgüte von Algorithmus D_1 und Algorithmus D_2

DS	Ergebnisgüte (A_2 besser ← → E_1 besser)	
A		0.6829
B		0.5521
C		0.1751
D		0.3500
E		0.2780
F		0.2162
G		0.2566
H		0.0798
I		0.1028
J		0.1979
K		0.1861
Ø		0.2798

(a) Produktionsdaten

DS	Ergebnisgüte (A_2 besser ← → E_1 besser)	
a		0.1167
b		0.1715
c		0.3182
d		0.3667
e		0.1151
f		0.2573
g		0.3634
h		0.5468
i		0.6691
j		0.2586
k		0.4197
l		0.5932
m		0.6823
n		0.7820
Ø		0.4043

(b) Zufällige Datensätze

TABELLE 5.15: Direkter Vergleich der Ergebnisgüte von Algorithmus A_2 und Algorithmus E_1. Für die zufälligen Datensätze o, p, q und r liegen aufgrund der langen Laufzeit von Algorithmus E_1 keine Ergebnisse vor.

Parameter	A	R	Wert
Name			E_1 (memetischer Algorithmus, Variante 1)
Populationsgröße			$\mu' = 0{,}1$
Anzahl der Kinder			$\lambda' = 1{,}15$
Selektionsstrategie			$(\mu + \lambda)$
Terminierung			Keine Verbesserung der besten Individuen ($\tau_{\text{last}} = 5$)
			Max. Anzahl an Auswertungen der Fitnessfunktion ($f_{\max} = 10000000$)
Initialisierung		1	Nächster Roboter & gierige Ersparnis-Heuristik (probabilistisch)
Paarungsselektion		1	Lineare rangbasierte Selektion ($c = 2$)
Umweltselektion		1	Stochastische Turnierselektion ($q = 5$)
Rekombination		1	Kantenrekombination ($p = 10$)
Mutation		1	Iterierte lokale Suche ($s = 1, w = 0{,}5$)

PARAMETERSATZ 5.13: Algorithmus E_1 (memetischer Algorithmus, Variante 1)

5.4.5 Memetische Algorithmen

Memetische Algorithmen (siehe Abschnitt 2.3.3.3) sind eine Kombination aus evolutionären Algorithmen und lokaler Suche. Wir stellen sie mit Hilfe der Kantenrekombination und des Mutationsoperators für die iterierte lokale Suche innerhalb unseres Ansatzes dar; Parametersatz 5.13 enthält die entsprechenden Einstellungen. Bis auf kleinere Änderungen handelt es sich um eine Kombination aus Parametersatz 5.2 (evolutionärer Algorithmus, Variante 2) und Parametersatz 5.12 (iterierte lokale Suche, Variante 2).

Gegenüber Algorithmus A_2 wurde die Anzahl der Kinder leicht verringert, um den Rechenaufwand zu reduzieren. Die Selektionsstrategie, die Initialisierung der Population und die Selektionsmethoden wurden von Algorithmus A_2 übernommen, die Terminierungsbedingungen und der Mutationsoperator von Algorithmus D_2. Die Identität tritt nicht mehr als Rekombinationsoperator auf. Die Anzahl der Elternindividuen des Kantenrekombinationsoperators wurde auf $p = 10$ erhöht, die Anzahl der Mutationsschritte zu Beginn der lokalen Suche auf $s = 1$ reduziert. Insgesamt entspricht die Parametereinstellung damit einem evolutionären Algorithmus (Rekombination, Mutation zu Beginn des Mutationsoperators), bei dem die Kindindividuen einem lokalen Suchverfahren unterworfen werden (im Iterationsschritt des Mutationsoperators für die iterierte lokale Suche, siehe Algorithmus 4.13).

Wir vergleichen Algorithmus E_1 in Tabelle 5.15 mit Algorithmus A_2 und in Tabelle 5.16 mit Algorithmus D_2. Auf den zufälligen Datensätzen o, p, q und r liegen für Algorithmus E_1 aufgrund der langen Laufzeit keine Ergebnisse vor (siehe Abschnitt 5.5.1). Dies liegt zum Teil daran, dass die Terminierungsbedingung „Maximale Anzahl an Auswertungen der Fitnessfunktion“ nur zu Beginn einer Generation überprüft wird, während bei jeder Ausführung des Mutationsoperators für die iterierte lokale Suche gemäß Satz 4.6 abhängig von der Problemgröße sehr viele Funktionsauswertungen vorgenommen werden.

DS	Ergebnisgüte (D_2 besser ← → E_1 besser)
A	0.0993
B	0.2026
C	0.0046
D	0.0274
E	0.0204
F	0,0000
G	0,0000
H	0.0009
I	0.0108
J	0.0069
K	0.0204
Ø	0.0317

(a) Produktionsdaten

DS	Ergebnisgüte (D_2 besser ← → E_1 besser)
a	0.0553
b	0.0235
c	0.0301
d	0.0844
e	0.0696
f	0.0514
g	0.0522
h	0.1275
i	0.1065
j	0.0952
k	0.0916
l	0.0609
m	0.0485
n	0.0673
Ø	0.0689

(b) Zufällige Datensätze

Tabelle 5.16: Direkter Vergleich der Ergebnisgüte von Algorithmus D_2 und Algorithmus E_1. Für die zufälligen Datensätze o, p, q und r liegen aufgrund der langen Laufzeit von Algorithmus E_1 keine Ergebnisse vor.

Während Algorithmus E_1 Algorithmus A_2 auf allen Datensätzen klar schlagen kann, bietet sich beim Vergleich mit Algorithmus D_2 das umgekehrte Bild, ausgenommen die beiden Datensätze F und G, auf denen beide Algorithmen das gleiche Ergebnis liefern. Der Vorsprung von Algorithmus D_2 gegenüber Algorithmus E_1 fällt jedoch geringer aus als derjenige von Algorithmus E_1 gegenüber Algorithmus A_2.

5.5 Gesamtauswertung

Während wir im vorherigen Abschnitt einzelne Parametersätze analysiert haben und Vergleiche auf wenige Parametersätze beschränkt waren, nehmen wir in diesem Abschnitt eine vergleichende Analyse aller Parametersätze vor. Wir gehen dazu kurz auf das vorliegende Datenmaterial und die Skalierung der Ergebnisse ein.

5.5.1 Auszuwertendes Datenmaterial

Für die Auswertung liegen die Daten aller durchgeführten Programmläufe der dreizehn in Abschnitt 5.4 definierten Algorithmen auf den zusammen 29 Datensätzen (elf Produktionsdatensätze und achtzehn zufällig erstellte Datensätze, siehe Abschnitt 5.2) vor.

Für jeden Algorithmus wurden, soweit möglich, 50 Programmläufe pro Datensatz durchgeführt, um genaue und zuverlässige Aussagen über das Verhalten der Algorithmen zu ermöglichen.[6] Für Algorithmus A_1 (Parametersatz 5.1) wurden auf Datensatz r nur 27 Programmläufe durchgeführt, für Algorithmus E_1 (Parametersatz 5.13) auf den Datensätzen o, p, q und r überhaupt keine. Ursächlich hierfür waren die langen Laufzeiten dieser Algorithmen auf den genannten Datensätzen und die begrenzte verfügbare Rechenzeit. Zusammen wurden 18 627 Programmläufe durchgeführt, wobei insgesamt 140 158 075 Generationen simuliert wurden.

5.5.2 Untere Schranken und Skalierung der Ergebnisse

Um die Ergebnisse von Programmläufen auf verschiedenen Datensätzen miteinander vergleichen zu können, skalieren wir diese mit Hilfe einer unteren Schranke für jeden Datensatz.

Da uns die Werte optimaler Lösungen nicht bekannt sind, verwenden wir das beste von allen Algorithmen in allen Programmläufen auf einem Datensatz erzielte Ergebnis als untere Schranke für diesen. Tabelle 5.17 zeigt diese Werte für alle Datensätze, zusammen mit dem schlechtesten Ergebnis für jeden Datensatz und den dazugehörigen Algorithmen.

Sei $\min_{\mathrm{d}} \in \mathbb{R}_{\geq 0}$, $\mathrm{d} \in \{A, \ldots, K, a, \ldots, r\}$ das beste in allen Programmläufen auf Datensatz d erzielte Ergebnis gemäß Tabelle 5.17. Das Ergebnis $-f_{\mathrm{best}}$ eines Programmlaufs auf diesem Datensatz kann dann vermöge

$$-f'_{\mathrm{best}} := \frac{-f_{\mathrm{best}} - \min_{\mathrm{d}}}{\min_{\mathrm{d}}} \in \mathbb{R}_{\geq 0}$$

skaliert werden. Der skalierte Ergebniswert kann als Prozentwert interpretiert werden. Er gibt an, um wieviel Prozent das Ergebnis $-f_{\mathrm{best}}$ über der unteren Schranke $\min_{\mathrm{d}}$ liegt.

5.5.3 Rangbasierte Bewertung der Algorithmen

Zur vergleichenden Bewertung aller Algorithmen bilden wir Ranglisten für diese. Der Rang eines Algorithmus hängt dabei vom Vergleich seiner skalierten Ergebnisse mit denjenigen der anderen Algorithmen ab. Wir gehen wie in Algorithmus 5.1 beschrieben vor. Beispiel 5.1 demonstriert das Verfahren für drei Algorithmen, vierzehn Programmläufe und zwei Datensätze.

[6] Für 50 Programmläufe pro Algorithmus und Datensatz beträgt die Größe des Konfidenzintervalls für den geschätzten Erwartungswert der Ergebnisse typischerweise weniger als 5% der Größe dieses geschätzten Erwartungswerts. Die genauen Werte sind den Tabellen in Anhang A zu entnehmen.

DS	bestes Ergebnis	Algorithmen (Anzahl Läufe)	schlechtestes Ergebnis	Alg.
a	2,78356	$B_2\,(50)$, $D_2\,(50)$, $D_1\,(49)$, $B_3\,(35)$, $E_1\,(5)$, $C_4\,(3)$, $B_1\,(2)$, $C_5\,(2)$	5,16446	$C_2\,(1)$
b	5,72055	$D_1\,(1)$	17,58467	$C_2\,(1)$
c	4,19493	$D_1\,(1)$	19,07442	$C_2\,(1)$
d	3,39324	$D_1\,(1)$	18,31663	$C_2\,(1)$
e	9,29090	$D_1\,(1)$	35,22721	$C_2\,(1)$
f	6,33211	$D_1\,(1)$	35,87907	$C_2\,(1)$
g	5,20247	$D_2\,(1)$	32,95405	$C_2\,(1)$
h	3,53268	$D_2\,(1)$	32,68132	$C_2\,(1)$
i	2,97940	$D_1\,(1)$	37,91216	$C_2\,(1)$
j	10,24137	$D_2\,(1)$	74,54211	$C_2\,(1)$
k	7,78575	$D_2\,(1)$	83,39057	$C_2\,(1)$
l	5,69882	$D_2\,(1)$	85,68107	$C_2\,(1)$
m	4,47107	$D_2\,(1)$	53,64289	$C_2\,(1)$
n	3,29362	$D_1\,(1)$	50,83999	$C_2\,(1)$
o	10,39790	$D_2\,(50)$	206,78300	$C_2\,(1)$
p	7,92567	$D_2\,(50)$	176,48558	$C_2\,(1)$
q	5,52281	$D_2\,(50)$	150,79202	$C_2\,(1)$
r	8,36895	$D_2\,(50)$	334,35329	$C_2\,(1)$

(b) Zufällige Datensätze

DS	bestes Ergebnis	Algorithmen (Anzahl Läufe)	schlechtestes Ergebnis	Alg.
A	10333,28	$D_2\,(1)$	91761,57	$C_2\,(1)$
B	13717,69	$D_1\,(1)$	194917,71	$C_2\,(1)$
C	5824,73	$E_1\,(21)$, $D_2\,(13)$, $B_2\,(7)$, $D_1\,(2)$	12187,13	$C_2\,(1)$
D	8471,24	$D_1\,(50)$, $D_2\,(50)$, $E_1\,(39)$	25827,97	$C_2\,(1)$
E	13130,38	$D_1\,(1)$	54054,90	$B_1\,(1)$
F	5175,06	$B_2\,(50)$, $B_3\,(50)$, $C_4\,(50)$, $C_5\,(50)$, $D_1\,(50)$, $D_2\,(50)$, $E_1\,(50)$, $A_1\,(30)$, $C_1\,(29)$, $B_1\,(25)$, $A_2\,(22)$, $C_3\,(15)$	9098,52	$C_2\,(1)$
G	5175,06	$B_2\,(50)$, $B_3\,(50)$, $D_1\,(50)$, $D_2\,(50)$, $E_1\,(50)$, $C_4\,(48)$, $C_5\,(46)$, $B_1\,(34)$, $C_1\,(25)$, $A_1\,(19)$, $A_2\,(16)$, $C_3\,(11)$	8848,32	$C_2\,(4)$
H	10943,29	$E_1\,(21)$, $B_2\,(8)$, $B_3\,(8)$, $D_1\,(6)$, $D_2\,(4)$, $C_5\,(2)$, $C_4\,(1)$	16533,98	$C_2\,(1)$
I	10966,05	$E_1\,(9)$, $B_3\,(2)$, $D_1\,(2)$, $B_2\,(1)$	17739,37	$C_2\,(1)$
J	6427,85	$D_2\,(13)$, $E_1\,(7)$	20394,08	$B_1\,(1)$
K	6972,74	$D_2\,(14)$, $E_1\,(13)$, $D_1\,(5)$, $B_2\,(4)$, $B_3\,(1)$	14205,58	$C_2\,(1)$

(a) Produktionsdaten

Tabelle 5.17: Obere und untere Schranken für die erzielten Ergebnisse auf allen Datensätzen. Für jedes Ergebnis sind die Algorithmen zusammen mit der Anzahl der Programmläufe angegeben, von bzw. in denen es erreicht wurde. Bei mehreren Angaben sind diese nach der Anzahl der Programmläufe und, bei gleicher Anzahl, nach den Kürzeln der Algorithmen sortiert.

Sei $D \subseteq \{A, \ldots, K, a, \ldots, r\}$ eine Menge von Datensätzen.

(1) Skaliere die Ergebnisse sämtlicher Programmläufe aller Algorithmen auf den Datensätzen in D gemäß Abschnitt 5.5.2 und markiere sie mit den dazugehörigen Algorithmen.

(2) Sortiere die skalierten Ergebnisse aufsteigend nach ihrem Wert.

(3) Weise jedem Ergebnis seinen Rang innerhalb der sortierten Folge von Ergebnissen zu; dabei erhalten gleiche Ergebnisse den gleichen Rang.

(4) Sortiere die Algorithmen aufsteigend nach dem Durchschnitt der ihnen zugeordneten Ränge.

ALGORITHMUS 5.1: Erstellung von Ranglisten

BEISPIEL 5.1 (RANGBASIERTE BEWERTUNG) Gegeben seien $D = \{E, a\}$, Algorithmus A_1 mit den Ergebnissen 17787,88, 17980,49 auf Datensatz E und 3,43, 3,13, 3,31 auf Datensatz a, Algorithmus B_1 mit den Ergebnissen 46184,42 auf Datensatz E und 2,97, 2,95 auf Datensatz a sowie Algorithmus E_1 mit den Ergebnissen 15416,36, 15567,93, 15257,26 auf Datensatz E und 3,11, 2,85, 2,85 auf Datensatz a.

In Schritt 1 skalieren wir die Ergebnisse gemäß Abschnitt 5.5.2 und merken uns für jedes Ergebnis den Algorithmus von dem es stammt. Wir erhalten für Datensatz E die Werte

$$(0{,}355, A_1), (0{,}369, A_1), (2{,}517, B_1), (0{,}174, E_1), (0{,}186, E_1), (0{,}162, E_1)$$

und für Datensatz a die Werte

$$(0{,}232, A_1), (0{,}124, A_1), (0{,}189, A_1), (0{,}067, B_1), (0{,}06, B_1), (0{,}117, E_1), (0{,}024, E_1), (0{,}024, E_1).$$

Dabei haben wir der Übersichtlichkeit halber die entstehenden Zahlen auf drei Nachkommastellen gerundet. In Schritt 2 sortieren wir die skalierten Ergebnisse aufsteigend nach ihrem Wert. Wir erhalten die Folge

$$\begin{array}{lllllll} (0{,}024, E_1), & (0{,}024, E_1), & (0{,}060, B_1), & (0{,}067, B_1), & (0{,}117, E_1), & (0{,}124, A_1), & (0{,}162, E_1), \\ (0{,}232, A_1), & (0{,}355, A_1), & (0{,}369, A_1), & (0{,}174, E_1), & (0{,}186, E_1), & (0{,}189, A_1), & (2{,}517, B_1). \end{array}$$

Anschließend ordnen wir in Schritt 3 jedem Ergebnis seinen Rang innerhalb der Folge zu, wobei gleiche Ergebnisse den gleichen Rang erhalten:

$$\begin{array}{lllll} (0{,}024, E_1) \leftarrow 1 & (0{,}024, E_1) \leftarrow 1 & (0{,}060, B_1) \leftarrow 2 & (0{,}067, B_1) \leftarrow 3 & (0{,}117, E_1) \leftarrow 4 \\ (0{,}124, A_1) \leftarrow 5 & (0{,}162, E_1) \leftarrow 6 & (0{,}232, A_1) \leftarrow 7 & (0{,}355, A_1) \leftarrow 8 & (0{,}369, A_1) \leftarrow 9 \\ (0{,}174, E_1) \leftarrow 10 & (0{,}186, E_1) \leftarrow 11 & (0{,}189, A_1) \leftarrow 12 & (2{,}517, B_1) \leftarrow 13. & \end{array}$$

Im letzten Schritt bilden wir zuerst die Durchschnitte über die Ränge der Algorithmen:

$$\begin{array}{ll} A_1 & \leftarrow \frac{1}{5}(5 + 7 + 8 + 9 + 12) = 8{,}2 \\ B_1 & \leftarrow \frac{1}{3}(2 + 3 + 13) = 6 \\ E_1 & \leftarrow \frac{1}{6}(1 + 1 + 4 + 6 + 10 + 11) = 5{,}5 \end{array}$$

Wir erhalten die Rangliste E_1, B_1, A_1. □

Für die Gesamtwertung der Algorithmen verwenden wir die in Tabelle 5.18 dargestellten drei Ranglisten, eine auf den Produktionsdaten, eine auf den zufälligen Datensätzen und eine auf allen Datensätzen zusammen. Die gezeigten Ranglisten lassen sich in drei Abschnitte unterteilen, so dass sich die Reihenfolge der Algorithmen nur innerhalb eines Abschnitts ändert:

	echte Datensätze		zufällige Datensätze		alle Datensätze	
Pos.	Alg.	gem. Rang	Alg.	gem. Rang	Alg.	gem. Rang
1.	D_2	183,389	D_2	326,006	D_2	616,975
2.	D_1	335,605	E_1	1090,070	E_1	1661,670
3.	E_1	451,807	D_1	1521,860	D_1	2045,080
4.	C_5	834,773	C_5	3422,650	C_5	4362,740
5.	B_2	1172,610	B_3	3926,970	B_3	5218,680
6.	B_3	1228,960	A_2	3953,060	C_4	5729,330
7.	C_4	1430,030	A_1	4126,330	A_2	5881,160
8.	A_1	1820,140	C_4	4361,000	A_1	5899,130
9.	A_2	1945,610	B_2	6459,380	B_2	7344,410
10.	C_1	2315,510	C_3	6828,570	C_3	8710,380
11.	C_3	2328,910	C_1	6980,500	C_1	8884,800
12.	B_1	2563,990	B_1	7929,950	B_1	10059,500
13.	C_2	3192,520	C_2	8383,350	C_2	10869,600

TABELLE 5.18: Ranglisten für alle Algorithmen, getrennt nach Produktionsdaten und zufälligen Datensätzen sowie einer Gesamtauswertung. Die Algorithmen sind nach ihrer Position in der jeweiligen Rangliste sortiert und mit ihrem gemittelten Rang innerhalb der Liste versehen.

Die vorderen Plätze eins bis vier, das Mittelfeld mit den Plätzen fünf bis neun und das Schlussfeld mit den Plätzen zehn bis dreizehn.

Auf den Plätzen zehn und elf im Schlussfeld befinden sich die beiden Algorithmen C_1 und C_3, gefolgt von B_1 auf dem vorletzten und C_2 auf dem letzten Platz. Die Platzierung dieser Algorithmen überrascht nicht, handelt es sich doch bei Algorithmus B_1 um einen auf das TSP zugeschnittenen Parametersatz und bei den Algorithmen C_1, C_2 und C_3 um die einfachsten Varianten des simulierten Abkühlens, deren Hauptaufgabe die Darstellung der Auswirkungen der Wahl des Parameters w war.

Im Mittelfeld ist das Bild uneinheitlicher. Die Algorithmen B_3, A_1, A_2 und C_4 weisen keine größeren Unterschiede bezüglich realer und zufälliger Datensätze in ihrer Platzierung auf. Algorithmus B_2 dagegen liegt bei den Produktionsdatensätzen auf Platz fünf, in der Gesamtwertung jedoch nur auf Platz neun. Wir deuten dies als einen Hinweis auf die Angepasstheit des Verfahrens an die Struktur realer Datensätze.

Auf den vorderen Plätzen herrschen klare Verhältnisse: Die erfolgreichste Variante des simulierten Abkühlens, Algorithmus C_5, liegt auf Platz vier, die beiden Algorithmen D_1 und E_1 belegen abwechselnd die Plätze zwei und drei. Den ersten Platz hat, auch mit einem deutlichen Vorsprung im gemittelten Rang, Algorithmus D_2 inne. Die gute Platzierung der Algorithmen D_1 und D_2 macht sich auch in Tabelle 5.17 bemerkbar, in der alle unteren Schranken (auch) von einem der beiden Algorithmen gefunden wurden.

Mit Blick auf die Wahl des Optimierungsverfahrens scheinen die evolutionären Algorithmen alleine keinen Vorteil zu bieten, ebensowenig die populationsbasierten Ansätze. Während das simulierte Abkühlen nach vergleichsweise aufwändiger Einstellung der Parameter recht gute Ergebnisse liefert, wird es von jeder Methode, die lokale Suchverfahren verwendet, deutlich übertroffen.

Ausschlaggebend für die gute Leistung der memetischen Algorithmen scheint damit nicht die Rekombination, sondern die lokale Optimierung der Individuen zu sein. Im Kontext der Platzierung von simuliertem Abkühlen und iterierter lokaler Suche halten wir eine gewisse Ähnlichkeit in der Vorgehensweise der beiden Optimierungsverfahren für beachtenswert: Beiden Algorithmen liegt ein Weg durch den Nachbarschaftsgraphen als zentrales Konzept zu Grunde. Die Unterschiede liegen hauptsächlich in der Bestimmung des nächsten Knotens. Während das simulierte Abkühlen die

Entscheidung stochastisch auf der Basis einer dynamischen Verteilung trifft, wird der Weg bei der iterierten lokalen Suche deterministisch bestimmt, unterbrochen von kurzen, rein zufälligen Wegstücken. Der Erfolg der beiden Verfahren legt die Untersuchung weiterer, auf der gleichen Idee basierender Verfahren wie z. B. der Tabu-Suche (siehe Abschnitt 2.3.3.3) nahe.

Zusammenfassend lässt sich sagen, dass die untersuchten Optimierungsverfahren deutliche Unterschiede bezüglich der Güte der erbrachten Ergebnisse aufweisen, wobei die Verfahren zum Teil ein unterschiedliches Verhalten auf den realen und auf den zufällig erzeugten Datensätzen zeigen. Als leistungsstärkstes Verfahren hat sich die iterierte lokale Suche erwiesen, gefolgt von den memetischen Algorithmen und dem simulierten Abkühlen.

Der Preis für die besseren Ergebnisse ist eine wesentlich größere Anzahl an Auswertungen der Fitnessfunktion. Tabelle 5.19 vergleicht die Anzahl der Funktionsauswertungen von Algorithmus C_5 mit der von Algorithmus D_1, Tabelle 5.20 die von Algorithmus D_1 und Algorithmus D_2.

Algorithmus D_1 benötigt auf den Produktionsdatensätzen im Durchschnitt etwa doppelt so viele Funktionsauswertungen wie Algorithmus C_5, auf den zufälligen Datensätzen etwa viermal so viele. Dagegen benötigt Algorithmus D_2 noch einmal das gut 200fache der Funktionsauswertungen von Algorithmus D_1 (auf den zufälligen Datensätzen sogar das 720fache, wenn man den Datensatz r mit berücksichtigt). Falls neben der Güte der Ergebnisse auch der benötigte Rechenaufwand eine Rolle spielt, sind die Algorithmen D_1 und C_5 weniger rechenintensive, dafür aber im Ergebnis schwächere Alternativen zu Algorithmus D_2.

Interessiert man sich für Algorithmen mit einer geringen Spannbreite der Ergebnisse, kann man den Tabellen in Anhang A Angaben zur Zuverlässigkeit (im Sinne einer niedrigen Varianz) der Algorithmen entnehmen.

	Funktionsauswertungen			
DS	Alg. C_5	Alg. D_1	Quotienten	
A	19096	31873	0,599	1,669
B	32646	57265	0,570	1,754
C	5682	7378	0,770	1,299
D	7463	5393	1,384	0,723
E	10953	22599	0,485	2,063
F	1248	4325	0,289	3,465
G	1231	4316	0,285	3,506
H	1605	5650	0,284	3,519
I	2150	8010	0,268	3,726
J	8464	11104	0,762	1,312
K	2664	9103	0,293	3,417
Ø	8473	15183	0,558	1,792

(a) Produktionsdaten

	Funktionsauswertungen			
DS	Alg. C_5	Alg. D_1	Quotienten	
a	1129	4423	0,255	3,917
b	5995	15553	0,385	2,594
c	6414	14938	0,429	2,329
d	6396	15655	0,409	2,448
e	13977	24594	0,568	1,760
f	14331	40885	0,351	2,853
g	13738	37273	0,369	2,713
h	12561	28545	0,440	2,272
i	12458	31679	0,393	2,543
j	25298	84881	0,298	3,355
k	25797	90709	0,284	3,516
l	21673	71349	0,304	3,292
m	22107	98238	0,225	4,444
n	19002	60092	0,316	3,162
o	44089	112159	0,393	2,544
p	37271	274978	0,136	7,378
q	27127	188533	0,144	6,950
r	12106	57437	0,211	4,744
Ø	17859	69551	0,257	3,894

(b) Zufällige Datensätze

Tabelle 5.19: Anzahl der Auswertungen der Fitnessfunktion für Algorithmus C_5 und Algorithmus D_1

5.6 Ausblick

Angesichts der Vielzahl an Parametern zur Kontrolle unseres evolutionären Ansatzes für das WCP-A wird klar, dass auch die vorgestellten dreizehn Parametersätze nur einen sehr kleinen Teil der möglichen Parametereinstellungen abdecken können. Bedenkt man die zum Teil komplexen Wechselwirkungen zwischen den einzelnen Parametern, liegt der Gedanke nahe, die Einstellung der Parameter automatisiert vornehmen zu lassen, d. h. die Einstellung der Parameter wieder als Optimierungsproblem aufzufassen.

Hierfür existieren mehrere Möglichkeiten: Die Kontrolle der Parameter innerhalb des Optimierungsprozesses selbst, entweder durch deterministische Regeln, vom Berechnungsverlauf abhängige Regeln (adaptive Verfahren) oder durch Evolution der Parameter (selbst-adaptive Verfahren), [56] sowie die Definition einer geeigneten Zielfunktion, um die Anwendung eines Optimierungsverfahrens zur Einstellung der Parameter zu ermöglichen. Findet die Einstellung der Parameter innerhalb des evolutionären Optimierungsprozesses selbst statt oder wird ein evolutionäres Verfahren zur Optimierung der Parameter verwendet, kann man von Metaevolution (Evolution der Parameter, d. h. der Bedingungen für den Evolutionsvorgang zur Lösung des eigentlichen Problems) sprechen.

Der von uns entworfene evolutionäre Ansatz für das WCP-A hat den Vorteil, dass sich die Parameter leicht algorithmisch manipulieren lassen; dies gilt sowohl für die Parameter der einzelnen Bauteile selbst, da es sich durchweg um mit einem Definitionsbereich versehene natürliche oder reelle Zahlen handelt, die leicht durch die üblichen Rekombinations- und Mutationsoperatoren

	Funktionsauswertungen			
DS	Alg. D_1	Alg. D_2	Quotienten	
A	31873	8173730	0,004	256,441
B	57265	12495653	0,005	218,204
C	7378	251615	0,029	34,100
D	5393	1832250	0,003	339,739
E	22599	6500997	0,003	287,659
F	4325	1707	2,533	0,395
G	4316	1819	2,372	0,422
H	5650	13734	0,411	2,431
I	8010	33917	0,236	4,234
J	11104	3328208	0,003	299,717
K	9103	266121	0,034	29,231
Ø	15183	2990886	0,005	196,978

(a) Produktionsdaten

	Funktionsauswertungen			
DS	Alg. D_1	Alg. D_2	Quotienten	
a	4423	7630	0,57968	1,725
b	15553	3118332	0,00499	200,497
c	14938	3559216	0,00420	238,255
d	15655	3268610	0,00479	208,777
e	24594	8165502	0,00301	332,004
f	40885	9073098	0,00451	221,917
g	37273	8546508	0,00436	229,290
h	28545	7785120	0,00367	272,727
i	31679	7503503	0,00422	236,860
j	84881	11086116	0,00766	130,607
k	90709	11756935	0,00772	129,610
l	71349	11883729	0,00600	166,558
m	98238	11743210	0,00837	119,537
n	60092	11745471	0,00512	195,455
o	112159	57618632	0,00195	513,718
p	274978	41141466	0,00668	149,617
q	188533	80124900	0,00235	424,989
r	57437	618585847	0,00009	10769,810
Ø	69551	50372990	0,00138	724,258

(b) Zufällige Datensätze

Tabelle 5.20: Anzahl der Auswertungen der Fitnessfunktion für Algorithmus D_1 und Algorithmus D_2. Berücksichtigt man Datensatz r nicht, beträgt der Durchschnitt des zweiten Quotienten in der Tabelle für zufällige Datensätze 241,21544.

für genetische Algorithmen manipuliert werden können, als auch für die Kombination mehrerer Methoden durch das einfache Schema absoluter und relativer Anforderungen in Algorithmus 4.2.

Eine Zielfunktion für die Bewertung eines Parametersatzes für Algorithmus 4.1 lässt sich wie folgt entwerfen: Man erstellt eine Menge von (gewichteten) Referenzdatensätzen (engl. benchmark) mit oberen und unteren Schranken und führt Algorithmus 4.1 mit den zu bewertenden Parametern eine festgelegte Anzahl $k \in \mathbb{N}$ von Malen auf jedem Datensatz aus, normiert die Ergebnisse mittels globaler Schranken und bildet den Durchschnitt über die normierten Ergebnisse. An Stelle des Durchschnitts über die Ergebnisse auf einem Datensatz kann man auch das schlechteste bzw. das beste Ergebnise pro Datensatz verwenden; denkbar ist auch die Berücksichtigung der Spanne zwischen schlechtestem und bestem Ergebnis der k Läufe auf einem Datensatz. Man optimiert in diesen Fällen nicht die durchschnittliche Leistung des Algorithmus, sondern den worst-case, den best-case bzw. die Zuverlässigkeit (Varianz). Durch die teure Auswertung der Zielfunktion ergeben sich andere Anforderungen an die Wahl des Optimierungsverfahrens (siehe Abschnitt 4.1).

Ein alternativer Ansatz zur Analyse des Algorithmus basiert auf einer genaueren Untersuchung der verwendeten Datensätze. Stehen genügend Produktionsdaten zur Verfügung, kann man versuchen, eine Verteilung auf diesen zu bestimmen, um anschließend eine average-case-Analyse einzelner Komponenten unter der Annahme dieser Verteilung durchzuführen. Eine solche Verteilung lässt sich auch zur Verbesserung der Bestandteile des Algorithmus ausnutzen, indem man diese auf die Struktur der Datensätze ausrichtet.

Kapitel 6

Ausblick & Fazit

6.1 Zusammenfassung

Mit der vorliegenden Arbeit wurde ein neues Problem aus dem Bereich der automatisierten Fertigung dem Kenntnisstand des Autors nach erstmals bearbeitet. Die Arbeit hatte die folgenden Zielsetzungen:

Definition und Modellierung Die Beschreibung und formale Definition des Problems. Die Arbeit enthält dazu eine ausführliche Beschreibung des Problems (Abschnitte 1.1, 1.2 und 1.3), auf deren Basis die formale Definition des Schweißzellenproblems in Abschnitt 1.4 erfolgt. In Abschnitt 3.3 werden die drei unterschiedlich stark vereinfachten Varianten WCP-A, WCP-B und WCP-C vorgestellt. Die Vereinfachung erfolgt auf systematische Art und Weise (Abschnitt 3.1).

Lösung des Problems Die algorithmische Bearbeitung des Problems einschließlich der Analyse. Zur Lösung der Variante WCP-A des Schweißzellenproblems wird ein flexibles evolutionäres Rahmenwerk entworfen (Abschnitt 4.4), innerhalb dessen mehrere stochastische Optimierungsverfahren abgebildet werden (Abschnitt 5.4). Kapitel 4 diskutiert zu diesem Zweck einige Aspekte evolutionärer Algorithmen (Abschnitte 4.2 und 4.3). Die Auswahl der Optimierungsverfahren wird begründet (Abschnitt 4.1). Die Verfahren werden auf echten und auf zufällig erzeugten Datensätzen (Abschnitt 5.2) empirisch analysiert (Abschnitte 5.3, 5.5 und Anhang A).

Arbeitsgrundlage Die Unterstützung bei der weiteren Beschäftigung mit dem Schweißzellenproblem. Dazu wird es in die relevanten Wissensgebiete Kollisionserkennung (Abschnitt 2.1), Ansteuerung der Roboter, Wegplanung (Abschnitt 2.2) und Optimierung (Abschnitt 2.3) eingeordnet. Für drei dieser Gebiete wird ein algorithmisch orientierter Überblick gegeben. Die Arbeit enthält Hinweise auf weiterführende Aufgabenstellungen (Abschnitte 1.5, 3.4, 4.4, 5.6 und 6.2), eine Aufstellung verwandter Probleme (Abschnitt 2.3.4) sowie ein kommentiertes, nach Sachgebieten geordnetes Literaturverzeichnis (ab Seite 139).

6.2 Ausblick

Im Anwendungsbereich des Schweißzellenproblems, der automatisierten Fertigung mit mehreren Industrierobotern, finden sich weitere eigenständige, interessante Problemstellungen mit Bezug zum Schweißzellenproblem. Ein Beispiel für ein solches Problem ist die Verriegelung von Robotern. Es handelt sich dabei um eine gängige Praxis zur Verhinderung von Kollisionen bei gegebenen

Bewegungen der Roboter, bei der man von diesen über die Bearbeitungszeit hinweg räumlich gemeinsam genutzte Gebiete identifiziert und entsprechende Verriegelungssignale setzt. Betritt ein Roboter ein solches Gebiet, setzt er das entsprechende Verriegelungssignal; verlässt er das Gebiet, setzt er das Signal wieder zurück. Versucht ein Roboter ein bereits verriegeltes Gebiet zu betreten, wird ein Nothalt ausgelöst. Insbesondere im Hinblick auf kurzfristige manuelle Anpassungen der Roboterbewegungen an Änderungen bei den Schweißpunkten und dem Werkstück ist die schnelle, automatisierte Bestimmung von korrekten Verriegelungssignalen bzw. die Überprüfung der Korrektheit bestehender Verriegelungsprogramme von Interesse.

Das Schweißzellenproblem selbst lässt sich sowohl als Optimierungs- als auch als Wegplanungsproblem auffassen. Während wir uns in dieser Arbeit auf den Optimierungsaspekt beschränkt haben, erfordert der Entwurf realistischerer Modelle die Berücksichtigung der Aspekte Kollisionserkennung und Wegplanung. Die in Abschnitt 3.3.3 vorgestellte Variante WCP-C kann dabei als Ausgangspunkt dienen. Sie bietet ein realistisches, praktisch unmittelbar verwertbares Modell um den Preis einer enormen Vergrößerung des Suchraums aufgrund der Einbeziehung der Wegplanung. Für Modelle dieser Art erscheint die Einbeziehung des in Abschnitt 2.2.4 vorgestellten Ansatzes der stochastischen Straßenkarten vielversprechend.

Weitere Möglichkeiten für die Bearbeitung der einfacheren Varianten des Schweißzellenproblems wie WCP-A und WCP-B sind die Anwendung von Techniken der mathematischen Programmierung (siehe Abschnitt 2.3.3.1) zur Bestimmung globaler Optima und die Bestimmung allgemeiner unterer Schranken, etwa auf Basis der Held-Karp-Schranke für das TSP. Die Aufstellung einer Sammlung von frei zugänglichen Referenzdatensätzen wäre ebenfalls hilfreich.

6.3 Fazit

Wir haben ein industriell relevantes Problem aus der automatisierten Fertigung mit Industrierobotern formal modelliert und die Lösbarkeit eines stark vereinfachten Modells mittels stochastischer Optimierungsverfahren demonstriert. Die vorgenommenen Vereinfachungen lassen eine unmittelbare Anwendbarkeit in der industriellen Praxis nicht zu; die Arbeit bietet jedoch einen Ausgangspunkt für die Untersuchung komplexerer Modelle.

Anhang A

Statistische Kenngrößen

Wir berechnen in diesem Anhang Schätzer für zwei statistische Kenngrößen, den Erwartungswert und die Varianz der Ergebnisse eines Algorithmus auf einem Datensatz sowie Konfidenzintervalle für diese. In den beiden folgenden Abschnitten wird die Berechnung der Schätzer und der Konfidenzintervalle erklärt; die numerischen Werte finden sich in den Tabellen A.1 bis A.13 am Ende des Anhangs.

A.1 Schätzer für Erwartungswert und Varianz

Der Wert[1] der von einem Algorithmus in einem Programmlauf gefundenen Lösung kann als eine Zufallsvariable mit Wertebereich $\mathbb{R}_{>0}$ interpretiert werden, die gemäß einer unbekannten Verteilung gezogen wird. Diese ist abhängig vom Datensatz und den Parametern des Algorithmus; ein Programmlauf entspricht einer Ziehung.

Um einschätzen zu können, wie oft ein Algorithmus auf einem Datensatz ausgeführt werden sollte, bis ein Ergebnis vorliegt, das von weiteren Läufen mit hoher Wahrscheinlichkeit nicht mehr übertroffen wird, ist die Kenntnis von Erwartungswert und Standardabweichung hilfreich. Da uns die Verteilung der Ergebnisse nicht bekannt ist, berechnen wir Schätzer für Erwartungswert und Varianz sowie Konfidenzintervalle zur Beurteilung dieser Schätzer. [79]

Wir fassen das Ergebnis eines einzelnen Programmlaufs als Ausgang eines Zufallsexperiments auf. Sei X die entsprechende Zufallsvariable und seien $x_1, \ldots, x_k$, $k \in \mathbb{N} \setminus \{1\}$ Realisierungen von X. Eine gute Schätzfunktion für einen Parameter ϑ eines stochastischen Modells oder einer Wahrscheinlichkeitsverteilung ist erwartungstreu (die Schätzfunktion hat den Erwartungswert ϑ), konsistent (die Schätzfunktion konvergiert mit zunehmendem Stichprobenumfang gegen ϑ) und effizient (keine andere erwartungstreue Schätzfunktion hat eine kleinere Varianz). Wir verwenden das arithmetische Mittel als erwartungstreuen, konsistenten und effizienten Schätzer für den Erwartungswert $\mathbb{E}[X]$: [79]

$$\hat{\mu}(x_1, \ldots, x_k) := \frac{1}{k} \sum_{i=1}^{k} x_i \ .$$

Für die Varianz $\mathbb{V}[X]$ verwenden wir die übliche, erwartungstreue und konsistente Schätzfunktion

$$\hat{\sigma}^2(x_1, \ldots, x_k) := \frac{1}{k-1} \sum_{i=1}^{k} \big(x_i - \hat{\mu}(x_1, \ldots, x_k)\big)^2 \ .$$

[1] Die folgenden Aussagen können in ihrer Art auch für die gefundenen Lösungen selbst getroffen werden; es ist jedoch einfacher, mit reellen Zahlen zu rechnen als mit angeordneten Partitionen.

A.2 Konfidenzintervalle

Ein *Konfidenzintervall* (auch Vertrauensintervall) für einen Schätzer $\hat{\vartheta}$ ist eine Zufallsgröße,[2] die den geschätzten Parameter ϑ mit einer vorgegebenen Wahrscheinlichkeit γ enthält. Wir berechnen Konfidenzintervalle für $\hat{\mu}$ und $\hat{\sigma}^2$ auf einem Konfidenzniveau von $\gamma = 0{,}95$. Für Details zur Anwendung und Berechnung von Konfidenzintervallen verweisen wir auf die Literatur [79].

Da genügend Stichproben (Programmläufe) vorliegen,[3] können wir bei der Berechnung der Konfidenzintervalle die Schätzfunktion $\hat{\mu}$ als normalverteilt annehmen. Die Vorgehensweise für die Berechnung eines Vertrauensintervalls für $\hat{\mu}$ ist bei gegebenem k, γ wie folgt:[4]

(1) Bestimmung von $c \in \mathbb{R}_{>0}$ mit $\mathbb{P}[-c \leq T \leq c] = \gamma$, wobei T der Student-Verteilung mit $k-1$ Freiheitsgraden genügt. Entsprechende Werte für c lassen sich z. B. durch eine Tabelle mit den Quantilen der Student-Verteilung oder mittels gängiger Mathematikprogramme bestimmen. Benötigt wird das Quantil für die Wahrscheinlichkeit $(1+\gamma)/2$.

(2) Berechnung der Grenzen des Konfidenzintervalls:

$$\mathbb{P}\left[\hat{\mu} - c\frac{\hat{\sigma}}{\sqrt{k}} \leq \mathbb{E}[X] \leq \hat{\mu} + c\frac{\hat{\sigma}}{\sqrt{k}}\right] = \gamma.$$

Zur Berechnung eines Konfidenzintervalls für $\hat{\sigma}^2$ geht man bei gegebenem k, γ so vor:

(1) Bestimmung von $c_1, c_2 \in \mathbb{R}_{>0}$ mit $\mathbb{P}[-c_1 \leq Z \leq c_2] = \gamma$, wobei Z der χ^2-Verteilung mit $k-1$ Freiheitsgraden genügt. Entsprechende Werte für c_1, c_2 lassen sich z. B. durch eine Tabelle mit den Quantilen der χ^2-Verteilung oder mittels gängiger Mathematikprogramme bestimmen. Benötigt werden die Quantile für die Wahrscheinlichkeiten $(1-\gamma)/2$ für c_1 und $(1+\gamma)/2$ für c_2.

(2) Berechnung der Grenzen des Konfidenzintervalls:

$$\mathbb{P}\left[\frac{(k-1)\hat{\sigma}^2}{c_2} \leq \mathbb{V}[X] \leq \frac{(k-1)\hat{\sigma}^2}{c_1}\right] = \gamma\ .$$

Die folgenden Tabellen A.1 bis A.13 enthalten pro Algorithmus und Datensatz das beste (kleinste) und das schlechteste (größte) berechnete Ergebnis, die in Abschnitt A.1 beschriebenen Schätzer $\hat{\mu}$ und $\hat{\sigma}$ für Erwartungswert und Varianz der Ergebnisse sowie Konfidenzintervalle für $\hat{\mu}$ und $\hat{\sigma}$ auf einem Konfidenzniveau von $\gamma = 0{,}95$.

[2] Die Berechnungsvorschrift für die Intervallgrenzen ist deterministisch, basiert aber auf $x_1, \ldots, x_k$ als Eingabe.

[3] Gemäß dem zentralen Grenzwertsatz der Wahrscheinlichkeitsrechnung konvergiert die Verteilung der Schätzfunktion gegen die Normalverteilung. In der Praxis sind Stichproben vom Umfang 30 oder höher für die Annahme der Normalverteilung ausreichend. [79] Für die Algorithmen liegen bis auf wenige Ausnahmen (siehe Abschnitt 5.5.1) 50 Programmläufe pro Datensatz vor.

[4] Bei unbekannter Varianz ist die Student-Verteilung zu verwenden; diese strebt für $k \to \infty$ gegen die Standardnormalverteilung. Bei genügend großen Stichproben kann daher in Schritt 1 die zu kleineren Konfidenzintervallen führende Standardnormalverteilung verwendet werden. Wir tun dies ab 27 Programmläufen pro Datensatz.

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
a	2,85	3,28	3,60	0,16	[3,23, 3,32]	[0,14, 0,20]
b	6,65	7,45	9,37	0,51	[7,31, 7,60]	[0,43, 0,63]
c	5,50	6,23	7,15	0,35	[6,13, 6,32]	[0,29, 0,43]
d	4,34	5,13	5,78	0,30	[5,04, 5,21]	[0,25, 0,37]
e	11,44	12,56	13,96	0,58	[12,40, 12,72]	[0,48, 0,71]
f	8,20	9,24	11,27	0,55	[9,08, 9,39]	[0,46, 0,68]
g	7,15	8,04	9,86	0,52	[7,89, 8,18]	[0,43, 0,64]
h	5,55	6,12	6,69	0,30	[6,04, 6,21]	[0,25, 0,37]
i	5,00	5,57	6,35	0,32	[5,48, 5,66]	[0,27, 0,40]
j	13,65	14,93	18,49	0,77	[14,72, 15,15]	[0,64, 0,95]
k	11,14	12,68	13,96	0,58	[12,52, 12,84]	[0,48, 0,71]
l	9,25	10,15	11,21	0,45	[10,02, 10,27]	[0,37, 0,55]
m	7,35	8,31	9,91	0,56	[8,15, 8,46]	[0,47, 0,69]
n	5,63	6,40	7,60	0,42	[6,28, 6,52]	[0,35, 0,52]
o	16,92	18,26	21,09	0,94	[18,00, 18,52]	[0,78, 1,16]
p	14,20	15,41	16,85	0,64	[15,23, 15,58]	[0,53, 0,79]
q	11,13	12,14	13,68	0,57	[11,98, 12,29]	[0,47, 0,70]
r	17,48	18,66	21,74	1,03	[18,27, 19,05]	[0,80, 1,38]

(b) Zufällige Datensätze

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
A	16684,93	18315,48	20461,76	723,85	[18114,84, 18516,12]	[599,65, 890,17]
B	21731,27	23774,06	27480,02	1347,47	[23400,56, 24147,56]	[1116,26, 1657,08]
C	6042,22	6733,04	7074,97	296,19	[6650,94, 6815,14]	[245,37, 364,25]
D	10523,53	11672,12	13077,20	545,19	[11521,00, 11823,24]	[451,65, 670,47]
E	16329,07	18509,82	21671,00	1063,46	[18215,05, 18804,60]	[880,99, 1307,82]
F	5175,06	5952,56	8742,33	1199,24	[5620,15, 6284,98]	[993,47, 1474,79]
G	5175,06	6103,20	8250,67	1203,14	[5769,71, 6436,69]	[996,70, 1479,59]
H	11033,69	11937,38	12539,90	449,91	[11812,67, 12062,09]	[372,71, 553,29]
I	11136,05	12115,57	12777,23	421,22	[11998,82, 12232,33]	[348,94, 518,00]
J	6896,64	7680,91	8477,64	307,75	[7595,61, 7766,21]	[254,94, 378,46]
K	7368,06	8486,59	9197,73	389,82	[8378,53, 8594,64]	[322,94, 479,39]

(a) Produktionsdaten

Tabelle A.1: Statistische Kenngrößen für Algorithmus A_1

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
a	2,92	3,26	3,60	0,16	[3,22, 3,31]	[0,13, 0,20]
b	6,74	7,34	8,07	0,28	[7,26, 7,41]	[0,23, 0,34]
c	5,27	6,22	6,64	0,27	[6,15, 6,30]	[0,22, 0,33]
d	4,66	5,25	5,73	0,24	[5,18, 5,31]	[0,20, 0,29]
e	11,18	11,85	12,52	0,29	[11,77, 11,93]	[0,24, 0,36]
f	8,07	8,77	9,40	0,29	[8,70, 8,85]	[0,24, 0,35]
g	7,26	7,83	8,49	0,24	[7,77, 7,90]	[0,20, 0,30]
h	5,78	6,24	6,87	0,24	[6,17, 6,30]	[0,20, 0,30]
i	5,05	5,63	6,01	0,23	[5,56, 5,69]	[0,19, 0,28]
j	13,54	13,93	14,45	0,21	[13,87, 13,98]	[0,18, 0,26]
k	11,35	12,03	12,69	0,31	[11,95, 12,12]	[0,26, 0,38]
l	9,40	9,86	10,56	0,24	[9,79, 9,92]	[0,20, 0,30]
m	7,63	8,03	8,53	0,23	[7,97, 8,10]	[0,19, 0,29]
n	5,95	6,36	6,66	0,18	[6,31, 6,40]	[0,15, 0,22]
o	16,02	16,64	16,98	0,20	[16,58, 16,69]	[0,16, 0,24]
p	13,75	14,41	14,86	0,21	[14,35, 14,47]	[0,18, 0,26]
q	11,11	11,61	12,05	0,24	[11,54, 11,67]	[0,20, 0,29]
r	16,72	17,18	17,66	0,21	[17,12, 17,24]	[0,18, 0,26]

(b) Zufällige Datensätze

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
A	17141,48	18737,49	20205,82	696,90	[18544,32, 18930,66]	[577,33, 857,03]
B	22680,04	24260,10	25475,45	651,16	[24079,61, 24440,59]	[539,43, 800,78]
C	6487,19	6893,53	6977,06	90,55	[6868,43, 6918,63]	[75,01, 111,35]
D	10549,46	11668,37	12151,13	379,92	[11563,06, 11773,67]	[314,73, 467,22]
E	17064,19	18722,97	20632,21	862,66	[18483,85, 18962,09]	[714,64, 1060,88]
F	5175,06	6293,75	8742,33	1320,87	[5927,62, 6659,87]	[1094,23, 1624,37]
G	5175,06	6502,79	8250,67	1299,58	[6142,57, 6863,02]	[1076,60, 1598,20]
H	11060,07	11914,33	12642,09	375,23	[11810,32, 12018,34]	[310,85, 461,45]
I	11328,23	12192,92	12946,61	367,11	[12091,16, 12294,68]	[304,12, 451,46]
J	7248,29	7746,32	8376,27	261,53	[7673,83, 7818,81]	[216,65, 321,62]
K	7363,42	8496,86	9772,95	474,20	[8365,42, 8628,30]	[392,84, 583,16]

(a) Produktionsdaten

Tabelle A.2: Statistische Kenngrößen für Algorithmus A_2

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
a	2,78	2,93	3,15	0,08	[2,90, 2,95]	[0,07, 0,10]
b	11,38	12,77	14,24	0,62	[12,60, 12,94]	[0,52, 0,77]
c	9,41	10,28	11,67	0,52	[10,14, 10,43]	[0,43, 0,63]
d	7,34	8,23	9,19	0,44	[8,11, 8,35]	[0,37, 0,54]
e	27,33	29,25	31,83	1,07	[28,95, 29,54]	[0,89, 1,32]
f	19,07	20,65	23,80	0,93	[20,39, 20,90]	[0,77, 1,14]
g	15,37	17,61	19,72	1,09	[17,30, 17,91]	[0,90, 1,34]
h	11,18	12,68	14,30	0,74	[12,48, 12,88]	[0,61, 0,90]
i	9,40	11,38	13,49	0,76	[11,17, 11,59]	[0,63, 0,93]
j	40,09	42,49	45,69	1,29	[42,13, 42,84]	[1,07, 1,59]
k	32,71	36,73	42,16	2,20	[36,12, 37,34]	[1,82, 2,71]
l	24,45	27,60	32,07	1,67	[27,14, 28,07]	[1,38, 2,05]
m	19,15	21,60	25,33	1,34	[21,23, 21,97]	[1,11, 1,65]
n	13,76	16,03	20,45	1,35	[15,65, 16,40]	[1,12, 1,66]
o	59,68	68,17	78,52	4,05	[67,05, 69,29]	[3,36, 4,98]
p	49,16	58,52	65,56	4,21	[57,35, 59,68]	[3,49, 5,18]
q	39,13	43,62	51,99	2,91	[42,81, 44,42]	[2,41, 3,57]
r	73,83	86,33	98,56	5,81	[84,72, 87,95]	[4,82, 7,15]

(b) Zufällige Datensätze

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
A	48136,28	58437,95	69853,67	4625,42	[57155,85, 59720,05]	[3831,77, 5688,22]
B	88478,19	103603,89	117593,91	5620,91	[102045,85, 105161,93]	[4656,45, 6912,45]
C	8141,88	9723,29	11838,72	912,13	[9470,46, 9976,12]	[755,62, 1121,71]
D	15358,77	20624,94	24880,69	2035,47	[20060,74, 21189,14]	[1686,22, 2503,17]
E	40902,89	47675,07	54054,90	2984,66	[46847,76, 48502,37]	[2472,54, 3670,45]
F	5175,06	5248,28	5825,64	149,13	[5206,94, 5289,61]	[123,54, 183,40]
G	5175,06	5197,36	5661,00	80,80	[5174,96, 5219,75]	[66,93, 99,36]
H	11041,87	11686,52	12305,61	289,95	[11606,15, 11766,89]	[240,20, 356,58]
I	11510,49	12361,27	13350,39	367,85	[12259,31, 12463,23]	[304,74, 452,38]
J	13556,46	16624,84	20394,08	1534,59	[16199,47, 17050,20]	[1271,28, 1887,20]
K	8124,96	10183,93	11957,67	869,56	[9942,90, 10424,96]	[720,36, 1069,36]

(a) Produktionsdaten

Tabelle A.3: Statistische Kenngrößen für Algorithmus B_1

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
a	2,78	2,78	2,78	0,00	[2,78, 2,78]	[0,00, 0,00]
b	6,22	6,96	8,43	0,54	[6,81, 7,11]	[0,45, 0,66]
c	5,06	6,03	7,49	0,58	[5,86, 6,19]	[0,48, 0,72]
d	4,18	5,21	6,53	0,55	[5,06, 5,36]	[0,45, 0,67]
e	11,19	15,54	22,41	2,71	[14,79, 16,29]	[2,24, 3,33]
f	10,50	13,83	16,81	1,68	[13,37, 14,30]	[1,39, 2,06]
g	10,05	12,32	14,72	1,03	[12,04, 12,61]	[0,85, 1,26]
h	7,65	8,75	10,07	0,56	[8,59, 8,90]	[0,47, 0,69]
i	6,72	7,64	8,81	0,43	[7,52, 7,75]	[0,36, 0,53]
j	29,56	33,22	38,30	1,98	[32,67, 33,77]	[1,64, 2,44]
k	25,15	27,97	31,31	1,63	[27,51, 28,42]	[1,35, 2,01]
l	18,09	19,96	22,26	1,01	[19,68, 20,24]	[0,84, 1,24]
m	13,89	15,44	17,43	0,92	[15,18, 15,69]	[0,76, 1,13]
n	10,15	11,09	13,86	0,88	[10,85, 11,34]	[0,73, 1,09]
o	47,82	52,63	59,43	2,92	[51,82, 53,44]	[2,42, 3,59]
p	37,32	44,10	51,17	2,93	[43,28, 44,91]	[2,43, 3,61]
q	28,42	32,62	39,73	2,80	[31,84, 33,40]	[2,32, 3,45]
r	56,91	65,06	74,20	4,08	[63,93, 66,19]	[3,38, 5,02]

(b) Zufällige Datensätze

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
A	21890,85	30376,01	40900,37	4533,65	[29119,34, 31632,67]	[3755,76, 5575,37]
B	51542,00	66793,25	84865,48	6249,85	[65060,88, 68525,62]	[5177,48, 7685,90]
C	5824,73	5940,19	6374,91	134,36	[5902,95, 5977,43]	[111,31, 165,23]
D	8488,37	9400,33	11707,71	786,73	[9182,26, 9618,41]	[651,74, 967,51]
E	14937,30	18288,08	25050,85	2644,87	[17554,96, 19021,20]	[2191,05, 3252,59]
F	5175,06	5175,06	5175,06	0,00	[5175,06, 5175,06]	[0,00, 0,00]
G	5175,06	5175,06	5175,06	0,00	[5175,06, 5175,06]	[0,00, 0,00]
H	10943,29	10988,57	11096,42	42,29	[10976,84, 11000,29]	[35,03, 52,01]
I	10966,05	11125,79	11400,61	109,04	[11095,56, 11156,01]	[90,33, 134,10]
J	6599,49	7173,44	8830,48	467,87	[7043,75, 7303,12]	[387,59, 575,38]
K	6972,74	7060,05	7449,32	84,49	[7036,63, 7083,47]	[69,99, 103,90]

(a) Produktionsdaten

Tabelle A.4: Statistische Kenngrößen für Algorithmus B_2

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
a	2,78	2,79	2,85	0,02	[2,78, 2,79]	[0,02, 0,02]
b	6,14	6,80	7,21	0,26	[6,72, 6,87]	[0,21, 0,32]
c	5,61	5,93	6,21	0,14	[5,89, 5,97]	[0,12, 0,18]
d	4,76	5,10	5,35	0,16	[5,05, 5,14]	[0,13, 0,19]
e	11,00	11,46	11,86	0,20	[11,41, 11,52]	[0,16, 0,24]
f	8,39	8,69	8,99	0,13	[8,66, 8,73]	[0,11, 0,16]
g	7,49	7,81	8,07	0,12	[7,78, 7,85]	[0,10, 0,15]
h	6,05	6,40	6,68	0,14	[6,37, 6,44]	[0,12, 0,17]
i	5,35	5,82	6,09	0,14	[5,78, 5,86]	[0,11, 0,17]
j	13,46	13,91	14,20	0,16	[13,86, 13,95]	[0,13, 0,20]
k	11,84	12,15	12,52	0,16	[12,11, 12,19]	[0,13, 0,20]
l	9,79	10,06	10,36	0,15	[10,02, 10,10]	[0,12, 0,18]
m	8,03	8,37	8,66	0,15	[8,33, 8,41]	[0,12, 0,18]
n	6,66	6,88	7,09	0,11	[6,86, 6,91]	[0,09, 0,13]
o	16,55	16,92	17,34	0,16	[16,88, 16,97]	[0,14, 0,20]
p	14,13	14,80	15,25	0,23	[14,73, 14,86]	[0,19, 0,28]
q	11,72	12,16	12,62	0,18	[12,11, 12,21]	[0,15, 0,22]
r	17,18	17,87	18,19	0,20	[17,81, 17,93]	[0,17, 0,25]

(b) Zufällige Datensätze

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
A	17546,23	18747,99	19475,07	444,42	[18624,81, 18871,18]	[368,17, 546,54]
B	23284,89	24628,57	25315,14	400,86	[24517,46, 24739,69]	[332,08, 492,97]
C	5858,72	6162,02	6827,35	318,00	[6073,88, 6250,17]	[263,43, 391,06]
D	9872,83	10511,88	10979,18	201,22	[10456,11, 10567,66]	[166,70, 247,46]
E	15465,90	16967,60	18168,68	651,24	[16787,09, 17148,12]	[539,50, 800,88]
F	5175,06	5175,06	5175,06	0,00	[5175,06, 5175,06]	[0,00, 0,00]
G	5175,06	5175,06	5175,06	0,00	[5175,06, 5175,06]	[0,00, 0,00]
H	10943,29	10997,84	11147,42	53,88	[10982,91, 11012,78]	[44,64, 66,26]
I	10966,05	11176,36	11605,58	130,01	[11140,33, 11212,40]	[107,70, 159,88]
J	7191,82	7421,87	7678,60	96,57	[7395,11, 7448,64]	[80,00, 118,76]
K	6972,74	7339,03	8185,87	238,51	[7272,92, 7405,14]	[197,58, 293,31]

(a) Produktionsdaten

Tabelle A.5: Statistische Kenngrößen für Algorithmus B_3

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
a	2,85	3,26	3,76	0,21	[3,20, 3,32]	[0,18, 0,26]
b	8,38	9,91	12,06	0,81	[9,68, 10,13]	[0,67, 1,00]
c	6,49	8,06	9,41	0,67	[7,88, 8,24]	[0,55, 0,82]
d	5,55	6,57	7,61	0,49	[6,43, 6,71]	[0,41, 0,60]
e	18,41	20,62	24,79	1,23	[20,28, 20,96]	[1,02, 1,51]
f	12,92	15,45	19,80	1,38	[15,06, 15,83]	[1,15, 1,70]
g	11,59	12,93	14,69	0,81	[12,71, 13,16]	[0,67, 1,00]
h	7,72	9,25	10,45	0,65	[9,07, 9,43]	[0,54, 0,80]
i	6,84	7,86	9,16	0,54	[7,71, 8,01]	[0,44, 0,66]
j	26,73	31,31	37,03	2,28	[30,68, 31,94]	[1,89, 2,80]
k	22,72	26,47	31,35	2,00	[25,92, 27,03]	[1,66, 2,46]
l	16,33	19,07	21,65	1,12	[18,76, 19,38]	[0,93, 1,38]
m	13,30	14,66	17,03	0,79	[14,44, 14,88]	[0,66, 0,97]
n	9,06	10,59	15,26	0,97	[10,32, 10,86]	[0,80, 1,19]
o	41,80	47,27	52,05	2,37	[46,61, 47,92]	[1,96, 2,91]
p	35,01	38,38	42,38	1,82	[37,88, 38,89]	[1,50, 2,23]
q	24,80	28,28	72,83	7,22	[26,27, 30,28]	[5,98, 8,88]
r	48,72	62,63	289,91	36,81	[52,43, 72,83]	[30,49, 45,26]

(b) Zufällige Datensätze

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
A	24912,33	31494,47	40183,55	3298,00	[30580,32, 32408,63]	[2732,12, 4055,79]
B	44850,09	56960,68	76275,62	7147,94	[54979,37, 58941,98]	[5921,47, 8790,35]
C	6203,62	7595,10	9719,72	738,93	[7390,28, 7799,92]	[612,14, 908,72]
D	11701,94	14332,89	17683,77	1368,15	[13953,65, 14712,12]	[1133,40, 1682,52]
E	24102,47	30646,94	39049,21	3273,27	[29739,64, 31554,25]	[2711,63, 4025,38]
F	5175,06	5794,68	8742,33	1005,60	[5515,94, 6073,42]	[833,06, 1236,67]
G	5175,06	5784,86	8304,87	1026,63	[5500,29, 6069,43]	[850,47, 1262,52]
H	11060,10	11805,29	12737,66	420,76	[11688,66, 11921,92]	[348,57, 517,45]
I	11511,46	12160,90	13164,79	382,24	[12054,95, 12266,85]	[316,65, 470,07]
J	8898,55	10526,26	12952,72	984,88	[10253,27, 10799,26]	[815,89, 1211,19]
K	7601,73	8915,66	11105,70	718,01	[8716,64, 9114,68]	[594,81, 882,99]

(a) Produktionsdaten

Tabelle A.6: Statistische Kenngrößen für Algorithmus C_1

	Kenngrößen				Konfidenzintervalle			
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$		$\hat{\sigma}$	
a	3,18	4,18	5,16	0,61	[4,01,	4,35]	[0,50,	0,75]
b	8,28	11,72	17,58	2,28	[11,09,	12,35]	[1,89,	2,80]
c	7,23	12,26	19,07	2,85	[11,47,	13,06]	[2,36,	3,51]
d	6,58	11,16	18,32	2,87	[10,37,	11,96]	[2,37,	3,52]
e	17,15	24,78	35,23	4,98	[23,39,	26,16]	[4,13,	6,13]
f	11,83	21,60	35,88	5,38	[20,10,	23,09]	[4,46,	6,62]
g	10,52	21,33	32,95	5,15	[19,90,	22,75]	[4,26,	6,33]
h	10,79	18,39	32,68	5,05	[16,99,	19,79]	[4,18,	6,21]
i	8,55	18,55	37,91	6,56	[16,74,	20,37]	[5,43,	8,06]
j	24,81	44,77	74,54	11,18	[41,67,	47,87]	[9,26,	13,75]
k	25,96	44,50	83,39	11,65	[41,27,	47,73]	[9,66,	14,33]
l	18,89	37,75	85,68	11,83	[34,47,	41,03]	[9,80,	14,55]
m	19,64	33,19	53,64	8,23	[30,91,	35,47]	[6,82,	10,13]
n	15,25	28,25	50,84	8,48	[25,90,	30,60]	[7,02,	10,42]
o	50,92	101,56	206,78	31,12	[92,93,	110,19]	[25,78,	38,27]
p	55,44	102,03	176,49	28,93	[94,01,	110,05]	[23,96,	35,57]
q	48,44	84,28	150,79	25,12	[77,32,	91,24]	[20,81,	30,89]
r	82,90	155,97	334,35	46,13	[143,18,	168,76]	[38,22,	56,73]

(b) Zufällige Datensätze

	Kenngrößen				Konfidenzintervalle			
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$		$\hat{\sigma}$	
A	29360,73	52766,64	91761,57	15711,74	[48411,57,	57121,71]	[13015,87,	19321,89]
B	52382,48	99807,37	194917,71	34497,47	[90245,16,	109369,58]	[28578,29,	42424,11]
C	7045,32	8676,62	12187,13	1213,28	[8340,32,	9012,93]	[1005,10,	1492,06]
D	12628,66	16443,76	25827,97	3101,45	[15584,08,	17303,44]	[2569,30,	3814,09]
E	23897,60	36096,54	50191,29	6436,58	[34312,41,	37880,67]	[5332,17,	7915,54]
F	5834,89	8087,66	9098,52	926,14	[7830,95,	8344,37]	[767,23,	1138,95]
G	6104,04	8177,89	8848,32	574,30	[8018,70,	8337,08]	[475,76,	706,27]
H	12348,77	14128,68	16533,98	929,95	[13870,91,	14386,45]	[770,39,	1143,63]
I	12986,36	14722,48	17739,37	1293,74	[14363,87,	15081,08]	[1071,76,	1591,01]
J	8463,71	12080,54	18398,07	2083,52	[11503,02,	12658,07]	[1726,03,	2562,26]
K	8692,21	10717,88	14205,58	1110,46	[10410,08,	11025,69]	[919,92,	1365,62]

(a) Produktionsdaten

TABELLE A.7: Statistische Kenngrößen für Algorithmus C_2

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
a	2,85	3,22	3,69	0,21	[3,16, 3,28]	[0,18, 0,26]
b	7,74	9,12	10,80	0,74	[8,91, 9,32]	[0,61, 0,91]
c	6,70	7,56	9,14	0,54	[7,41, 7,71]	[0,45, 0,66]
d	5,55	6,30	7,27	0,40	[6,19, 6,41]	[0,33, 0,49]
e	15,85	18,70	26,85	1,82	[18,20, 19,21]	[1,51, 2,24]
f	11,83	14,00	17,35	1,16	[13,68, 14,32]	[0,96, 1,43]
g	10,46	12,18	14,54	0,99	[11,91, 12,46]	[0,82, 1,22]
h	7,89	9,09	12,43	0,77	[8,88, 9,30]	[0,64, 0,95]
i	6,67	8,05	18,98	1,74	[7,57, 8,54]	[1,44, 2,14]
j	23,10	27,96	34,50	2,38	[27,29, 28,62]	[1,97, 2,93]
k	20,20	23,87	27,86	1,65	[23,41, 24,33]	[1,37, 2,03]
l	15,42	18,59	24,94	1,59	[18,15, 19,03]	[1,32, 1,95]
m	12,14	14,96	28,07	2,58	[14,25, 15,68]	[2,14, 3,17]
n	9,57	13,70	36,96	5,65	[12,13, 15,26]	[4,68, 6,95]
o	41,62	45,69	51,26	2,64	[44,96, 46,42]	[2,19, 3,25]
p	32,72	37,19	42,75	2,10	[36,61, 37,77]	[1,74, 2,58]
q	25,07	41,72	77,68	15,47	[37,43, 46,01]	[12,81, 19,02]
r	50,57	98,84	262,11	44,51	[86,51, 111,18]	[36,87, 54,74]

(b) Zufällige Datensätze

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
A	23494,77	29938,30	39888,76	3696,05	[28913,81, 30962,80]	[3061,87, 4545,31]
B	42150,67	51751,68	73433,86	7002,94	[49810,57, 53692,80]	[5801,36, 8612,04]
C	6500,64	7278,29	8347,59	392,98	[7169,36, 7387,22]	[325,56, 483,28]
D	11850,41	13505,05	16143,49	1016,73	[13223,23, 13786,88]	[842,28, 1250,35]
E	22644,49	26287,10	33838,66	2760,91	[25521,82, 27052,38]	[2287,18, 3395,29]
F	5175,06	6081,83	8335,07	1128,21	[5769,10, 6394,55]	[934,63, 1387,45]
G	5175,06	6199,54	8250,67	1129,38	[5886,49, 6512,59]	[935,60, 1388,88]
H	11041,87	11986,44	13026,09	477,35	[11854,13, 12118,76]	[395,44, 587,03]
I	11376,83	12410,41	13096,41	363,67	[12309,60, 12511,22]	[301,27, 447,24]
J	7910,45	9352,65	10956,05	685,67	[9162,60, 9542,71]	[568,02, 843,22]
K	7620,13	8831,44	9649,46	487,19	[8696,40, 8966,48]	[403,59, 599,13]

(a) Produktionsdaten

Tabelle A.8: Statistische Kenngrößen für Algorithmus C_3

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
a	2,78	3,19	3,60	0,22	[3,13, 3,25]	[0,18, 0,27]
b	5,99	7,13	8,31	0,42	[7,02, 7,25]	[0,35, 0,52]
c	4,94	5,75	6,69	0,38	[5,65, 5,86]	[0,31, 0,47]
d	4,19	4,71	5,60	0,30	[4,62, 4,79]	[0,25, 0,37]
e	11,22	12,50	14,50	0,66	[12,32, 12,68]	[0,54, 0,81]
f	8,01	8,88	10,56	0,55	[8,73, 9,03]	[0,45, 0,67]
g	6,39	7,51	9,54	0,50	[7,37, 7,64]	[0,41, 0,61]
h	4,99	5,65	6,72	0,37	[5,55, 5,76]	[0,31, 0,46]
i	4,31	4,94	5,82	0,37	[4,84, 5,04]	[0,31, 0,45]
j	14,50	16,19	19,17	1,05	[15,90, 16,48]	[0,87, 1,29]
k	11,76	13,30	15,68	0,92	[13,04, 13,55]	[0,77, 1,14]
l	9,17	10,48	12,09	0,72	[10,29, 10,68]	[0,59, 0,88]
m	7,41	8,44	10,28	0,54	[8,29, 8,59]	[0,45, 0,66]
n	5,60	6,52	7,96	0,55	[6,36, 6,67]	[0,46, 0,68]
o	20,69	23,93	28,24	1,51	[23,51, 24,35]	[1,25, 1,86]
p	17,62	20,18	23,22	1,37	[19,81, 20,56]	[1,13, 1,68]
q	13,84	15,69	19,02	1,18	[15,37, 16,02]	[0,98, 1,45]
r	27,12	30,74	36,00	1,90	[30,22, 31,27]	[1,57, 2,33]

(b) Zufällige Datensätze

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
A	16092,96	17472,49	20278,59	751,19	[17264,27, 17680,71]	[622,30, 923,79]
B	21292,01	24067,35	27038,94	1282,54	[23711,84, 24422,85]	[1062,47, 1577,23]
C	5921,10	6534,34	7029,82	337,96	[6440,66, 6628,02]	[279,97, 415,61]
D	10424,60	11167,08	11837,29	355,92	[11068,43, 11265,74]	[294,85, 437,70]
E	15910,19	17541,99	19423,13	773,36	[17327,62, 17756,35]	[640,67, 951,06]
F	5175,06	5175,06	5175,06	0,00	[5175,06, 5175,06]	[0,00, 0,00]
G	5175,06	5271,31	7586,17	476,32	[5139,29, 5403,34]	[394,59, 585,76]
H	10943,29	11462,74	12453,71	412,43	[11348,42, 11577,06]	[341,66, 507,19]
I	11042,82	11619,51	12552,59	452,85	[11493,99, 11745,03]	[375,15, 556,90]
J	6666,28	7396,44	7929,69	232,98	[7331,86, 7461,02]	[193,01, 286,51]
K	7152,02	7972,07	8641,06	441,42	[7849,71, 8094,42]	[365,68, 542,85]

(a) Produktionsdaten

Tabelle A.9: Statistische Kenngrößen für Algorithmus C_4

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
a	2,78	3,02	3,12	0,12	[2,99, 3,05]	[0,10, 0,14]
b	6,06	6,92	7,64	0,35	[6,82, 7,02]	[0,29, 0,43]
c	4,86	5,29	5,36	0,13	[5,25, 5,32]	[0,11, 0,16]
d	4,08	4,53	4,83	0,21	[4,48, 4,59]	[0,18, 0,26]
e	10,75	11,28	11,35	0,15	[11,24, 11,32]	[0,12, 0,18]
f	7,24	8,47	9,42	0,42	[8,35, 8,58]	[0,35, 0,52]
g	6,26	7,30	7,95	0,48	[7,17, 7,43]	[0,40, 0,59]
h	4,79	5,22	5,31	0,14	[5,18, 5,26]	[0,12, 0,18]
i	4,23	4,47	4,49	0,06	[4,46, 4,49]	[0,05, 0,07]
j	13,76	15,05	15,58	0,57	[14,89, 15,21]	[0,48, 0,71]
k	11,18	12,12	12,24	0,26	[12,05, 12,19]	[0,21, 0,32]
l	8,24	8,37	8,38	0,03	[8,36, 8,37]	[0,02, 0,03]
m	6,93	8,00	8,28	0,34	[7,91, 8,09]	[0,28, 0,42]
n	4,84	4,87	4,87	0,01	[4,87, 4,87]	[0,00, 0,01]
o	16,38	16,53	16,54	0,02	[16,52, 16,54]	[0,02, 0,03]
p	17,10	17,98	18,10	0,25	[17,91, 18,05]	[0,21, 0,31]
q	10,14	10,25	10,27	0,03	[10,25, 10,26]	[0,02, 0,04]
r	19,35	19,46	19,48	0,03	[19,45, 19,47]	[0,03, 0,04]

(b) Zufällige Datensätze

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
A	12345,62	12346,00	12346,00	0,05	[12345,98, 12346,01]	[0,05, 0,07]
B	16279,70	16423,11	16426,06	20,69	[16417,37, 16428,85]	[17,14, 25,45]
C	5893,00	6100,30	6143,06	78,91	[6078,43, 6122,18]	[65,37, 97,05]
D	8472,43	8472,43	8472,43	0,00	[8472,43, 8472,43]	[0,00, 0,00]
E	16145,47	16467,01	16490,75	73,58	[16446,62, 16487,41]	[60,96, 90,49]
F	5175,06	5175,06	5175,06	0,00	[5175,06, 5175,06]	[0,00, 0,00]
G	5175,06	5200,27	5490,18	86,36	[5176,34, 5224,21]	[71,54, 106,20]
H	10943,29	11401,93	12089,36	303,34	[11317,85, 11486,01]	[251,29, 373,04]
I	11011,66	11584,02	12453,71	484,79	[11449,64, 11718,40]	[401,61, 596,18]
J	6688,48	7275,96	7398,21	189,95	[7223,31, 7328,61]	[157,36, 233,60]
K	7031,98	8019,75	8794,49	467,57	[7890,14, 8149,35]	[387,34, 575,01]

(a) Produktionsdaten

Tabelle A.10: Statistische Kenngrößen für Algorithmus C_5

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
a	2,78	2,79	2,85	0,01	[2,78, 2,79]	[0,01, 0,01]
b	5,72	6,18	6,66	0,21	[6,13, 6,24]	[0,17, 0,25]
c	4,19	4,63	5,23	0,20	[4,57, 4,68]	[0,16, 0,24]
d	3,39	3,83	4,27	0,19	[3,78, 3,88]	[0,16, 0,23]
e	9,29	9,99	10,51	0,27	[9,92, 10,07]	[0,23, 0,33]
f	6,33	6,88	7,81	0,33	[6,79, 6,97]	[0,27, 0,40]
g	5,36	5,90	6,47	0,25	[5,83, 5,97]	[0,20, 0,30]
h	3,77	4,13	4,79	0,22	[4,06, 4,19]	[0,18, 0,27]
i	2,98	3,50	4,03	0,24	[3,43, 3,56]	[0,20, 0,30]
j	10,37	11,50	13,19	0,72	[11,30, 11,70]	[0,59, 0,88]
k	8,11	8,81	10,22	0,43	[8,69, 8,93]	[0,36, 0,53]
l	5,91	6,55	7,84	0,35	[6,45, 6,65]	[0,29, 0,44]
m	4,84	5,42	6,48	0,33	[5,32, 5,51]	[0,27, 0,41]
n	3,29	3,89	4,28	0,22	[3,83, 3,95]	[0,18, 0,27]
o	11,74	14,12	16,07	0,98	[13,85, 14,39]	[0,81, 1,21]
p	9,66	11,24	14,99	0,99	[10,96, 11,51]	[0,82, 1,22]
q	6,81	7,94	9,40	0,64	[7,77, 8,12]	[0,53, 0,78]
r	16,94	18,07	19,04	0,51	[17,93, 18,21]	[0,42, 0,63]

(b) Zufällige Datensätze

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
A	10495,77	11383,60	12264,37	427,79	[11265,03, 11502,18]	[354,39, 526,08]
B	13717,69	15113,53	16404,29	700,30	[14919,41, 15307,64]	[580,14, 861,22]
C	5824,73	5911,39	5946,57	43,88	[5899,23, 5923,56]	[36,35, 53,96]
D	8471,24	8471,24	8471,24	0,00	[8471,24, 8471,24]	[0,00, 0,00]
E	13130,38	14259,23	15612,84	714,55	[14061,17, 14457,30]	[591,95, 878,74]
F	5175,06	5175,06	5175,06	0,00	[5175,06, 5175,06]	[0,00, 0,00]
G	5175,06	5175,06	5175,06	0,00	[5175,06, 5175,06]	[0,00, 0,00]
H	10943,29	11074,90	11482,19	130,90	[11038,61, 11111,18]	[108,44, 160,98]
I	10966,05	11142,83	11584,36	136,96	[11104,87, 11180,80]	[113,46, 168,43]
J	6516,81	6680,08	7143,23	112,65	[6648,85, 6711,30]	[93,32, 138,54]
K	6972,74	7387,15	8015,12	237,30	[7321,37, 7452,93]	[196,58, 291,82]

(a) Produktionsdaten

Tabelle A.11: Statistische Kenngrößen für Algorithmus D_1

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
a	2,78	2,78	2,78	0,00	[2,78, 2,78]	[0,00, 0,00]
b	5,81	6,22	6,63	0,17	[6,17, 6,27]	[0,14, 0,20]
c	4,42	4,76	5,04	0,15	[4,72, 4,80]	[0,12, 0,18]
d	3,62	3,72	3,72	0,02	[3,71, 3,72]	[0,01, 0,02]
e	9,70	10,14	10,20	0,12	[10,10, 10,17]	[0,10, 0,14]
f	6,42	6,82	6,98	0,13	[6,78, 6,86]	[0,11, 0,16]
g	5,20	5,67	6,05	0,18	[5,62, 5,72]	[0,15, 0,22]
h	3,53	3,85	3,88	0,07	[3,84, 3,87]	[0,06, 0,08]
i	3,13	3,32	3,46	0,10	[3,29, 3,34]	[0,08, 0,12]
j	10,24	10,30	10,30	0,01	[10,30, 10,30]	[0,01, 0,01]
k	7,79	8,05	8,06	0,04	[8,04, 8,06]	[0,03, 0,05]
l	5,70	6,13	6,23	0,15	[6,09, 6,17]	[0,12, 0,18]
m	4,47	4,76	4,92	0,14	[4,73, 4,80]	[0,12, 0,17]
n	3,33	3,56	3,60	0,07	[3,54, 3,58]	[0,06, 0,09]
o	10,40	10,40	10,40	0,00	[10,40, 10,40]	[0,00, 0,00]
p	7,93	7,93	7,93	0,00	[7,93, 7,93]	[0,00, 0,00]
q	5,52	5,52	5,52	0,00	[5,52, 5,52]	[0,00, 0,00]
r	8,37	8,37	8,37	0,00	[8,37, 8,37]	[0,00, 0,00]

(b) Zufällige Datensätze

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
A	10333,28	10654,90	10686,98	80,09	[10632,70, 10677,09]	[66,34, 98,49]
B	13757,16	13907,10	13910,16	21,64	[13901,10, 13913,10]	[17,92, 26,61]
C	5824,73	5900,52	6048,34	79,22	[5878,56, 5922,48]	[65,63, 97,42]
D	8471,24	8471,24	8471,24	0,00	[8471,24, 8471,24]	[0,00, 0,00]
E	13192,09	14804,77	15327,72	518,66	[14661,01, 14948,54]	[429,66, 637,83]
F	5175,06	5175,06	5175,06	0,00	[5175,06, 5175,06]	[0,00, 0,00]
G	5175,06	5175,06	5175,06	0,00	[5175,06, 5175,06]	[0,00, 0,00]
H	10943,29	11031,14	11060,10	48,27	[11017,76, 11044,53]	[39,99, 59,37]
I	10991,79	11183,61	11282,52	109,02	[11153,39, 11213,83]	[90,32, 134,07]
J	6427,85	6518,11	7025,16	111,96	[6487,07, 6549,14]	[92,75, 137,69]
K	6972,74	7056,87	7194,83	80,60	[7034,53, 7079,22]	[66,77, 99,12]

(a) Produktionsdaten

Tabelle A.12: Statistische Kenngrößen für Algorithmus D_2

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
a	2,78	2,94	3,31	0,12	[2,90, 2,97]	[0,10, 0,15]
b	6,01	6,35	6,71	0,14	[6,32, 6,39]	[0,12, 0,17]
c	4,56	4,89	5,19	0,12	[4,85, 4,92]	[0,10, 0,15]
d	3,76	4,00	4,26	0,11	[3,97, 4,03]	[0,09, 0,14]
e	10,23	10,78	11,15	0,18	[10,73, 10,83]	[0,15, 0,22]
f	6,66	7,15	7,63	0,20	[7,09, 7,20]	[0,17, 0,25]
g	5,59	5,94	6,56	0,18	[5,89, 5,99]	[0,15, 0,22]
h	3,93	4,31	4,55	0,12	[4,27, 4,34]	[0,10, 0,15]
i	3,41	3,63	3,83	0,10	[3,60, 3,66]	[0,08, 0,12]
j	10,64	11,28	11,57	0,19	[11,23, 11,33]	[0,16, 0,23]
k	8,26	8,76	9,05	0,18	[8,71, 8,81]	[0,15, 0,22]
l	6,09	6,48	6,75	0,16	[6,43, 6,52]	[0,13, 0,20]
m	4,77	4,98	5,24	0,11	[4,95, 5,01]	[0,09, 0,14]
n	3,59	3,78	3,91	0,08	[3,76, 3,80]	[0,07, 0,10]

(b) Zufällige Datensätze

	Kenngrößen				Konfidenzintervalle	
DS	min	$\hat{\mu}$	max	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
A	10522,61	11681,28	13564,79	671,41	[11495,17, 11867,38]	[556,21, 825,68]
B	14573,45	16685,99	17730,55	546,28	[16534,57, 16837,41]	[452,55, 671,81]
C	5824,73	5873,78	5958,17	59,76	[5857,21, 5890,34]	[49,51, 73,49]
D	8471,24	8703,55	9858,59	475,80	[8571,67, 8835,44]	[394,16, 585,13]
E	13913,83	15072,08	15986,44	462,65	[14943,84, 15200,32]	[383,26, 568,95]
F	5175,06	5175,06	5175,06	0,00	[5175,06, 5175,06]	[0,00, 0,00]
G	5175,06	5175,06	5175,06	0,00	[5175,06, 5175,06]	[0,00, 0,00]
H	10943,29	11041,08	11740,95	165,64	[10995,17, 11086,99]	[137,22, 203,71]
I	10966,05	11065,34	11282,52	71,55	[11045,51, 11085,17]	[59,27, 87,99]
J	6427,85	6473,97	6628,06	50,90	[6459,86, 6488,08]	[42,17, 62,60]
K	6972,74	7199,03	7410,29	170,21	[7151,85, 7246,21]	[141,00, 209,32]

(a) Produktionsdaten

TABELLE A.13: Statistische Kenngrößen für Algorithmus E_1.
Für die Datensätze o, p, q, r liegen keine Programmläufe vor.

Literaturverzeichnis

Kollisionserkennung

Überblick & Allgemeines

[1] Ming Lin und Stefan Gottschalk: *Collision Detection Between Geometric Models: A Survey.* In *IMA Conference on Mathematics of Surfaces*, Birmingham, England, August & September 1998, Seite 37–56.

Der Artikel gibt eine Übersicht über Fragestellungen der Kollisionserkennung, Repräsentationsformen dreidimensionaler Modelle, Methoden zur Kollisionserkennung sowie frei verfügbare Programmpakete zur Kollisionserkennung.

[2] Gabriel Zachmann und Elmar Langetepe: *Geometric Data Structures for Computer Graphics.* Tutorium, *30th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 03)*, San Diego, Kalifornien, USA, 27.–31. Juli 2003, `www.gabrielzachmann.org` → Tutorials/Talks.

Vorstellung (Definition, Eigenschaften, Algorithmen, Anwendungen) von praktischen Datenstrukturen aus der rechenbetonten Geometrie. Themen sind u. a. Octrees, *K-d*-Bäume, BSP-Bäume, Hüllvolumen-Hierarchien, Voronoi-Diagramme, Distanzfelder und die Dynamisierung geometrischer Datenstrukturen.

[3] David Eberly: *3D Game Engine Design. A Practical Approach to Real-Time Computer Graphics*, 2001, Morgan Kaufman.

Eine umfassende Einführung in die Echtzeitdarstellung von Computergrafik. In unserem Zusammenhang besonders interessant ist das Kapitel über geometrische Methoden (grundlegende geometrische Objekte, Koordinatensysteme, Winkel, Distanzberechnung) sowie die beiden Kapitel Picking und Kollisionserkennung (Schnitt geometrischer Objekte, sowohl statisch als auch dynamisch). Für fortgeschrittene Methoden der Kollisionserkennung und der Wegplanung sind auch das Kapitel über räumliche Sortierung und der Abschnitt über inverse Kinematik relevant.

Hüllvolumen-Ansätze

[4] Ming Lin, Dinesh Manocha, Jon Cohen und Stefan Gottschalk: *Collision Detection: Algorithms and Applications.* In Jean-Paul Laumond und Mark Overmars (Herausgeber): *Algorithms for Robot Motion and Manipulation*, 1996, A. K. Peters, Seite 129–142.

Beginnt mit einem kurzen Überblick über das Gebiet der Kollisionserkennung und ihre Anwendungen. Anschließend wird ein Algorithmus vorgestellt, der auf einer hierarchischen Repräsentation mit orientierten Quadern, Voronoi-Regionen und Projektion auf die Achsen beruht. Mit experimentellen Ergebnissen.

[5] Stefan Gottschalk, Ming Lin und Dinesh Manocha: *OBBTree: A Hierarchical Structure for Rapid Interference Detection.* In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (Siggraph 96)*, New Orleans, Louisiana, USA, 4.–9. August 1996, Seite 171–180, ACM.

Beschreibt einen Algorithmus zur Kollisionserkennung auf der Basis von orientierten Quadern als Hüllvolumen. Tests auf Kollision zweier orientierter Quader erfolgen mittels eines Separationsachsen-Theorems. Enthält einen kurzen Überblick über Methoden zur Kollisionserkennung, einen analytischen Vergleich von achsenausgerichteten Quadern und Kugeln als Hüllvolumen und experimentelle Ergebnisse. Errata sind unter `www.cs.unc.edu/~geom/OBB/OBBT.html` → Paper zu finden.

[6] Stefan Gottschalk: *Collision Queries using Oriented Bounding Boxes.* Dissertation, 2000, University of North Carolina at Chapel Hill.

Enthält u. a. ein allgemeines Kapitel über Hüllvolumen-Bäume sowie Kapitel über orientierte Quader als Hüllvolumen, wobei insbesondere auf die Berechnung von möglichst gut angepassten OBBs (engl. oriented bounding box) und auf Kollisionstests für diese eingegangen wird. Die Arbeit enthält einen Überblick über Verfahren zur Kollisionserkennung. Mit experimentellen Ergebnissen.

[7] Thomas Hudson, Ming Lin, Jonathan Cohen, Stefan Gottschalk und Dinesh Manocha: *V-COLLIDE: Accelerated Collision Detection for VRML.* In *Proceedings of the Second Symposium on Virtual Reality Modeling Language (VRML 97)*, Monterey, Kalifornien, USA, 24.–26. Februar 1997, Seite 117–123, ACM.

Der Artikel beschreibt das V-Collide-System zur Kollisionserkennung bei mehreren Objekten im Kontext der VRML-Spezifikationssprache. Es wird auf den Aufbau von V-Collide (n-Körper-Test mittels achsenausgerichteter Quader, Hierarchie von orientierten Quadern als Hüllvolumen) sowie auf die Anwendbarkeit und die Integration in VRML eingegangen.

[8] Pierre Terdiman: *Memory-Optimized Bounding-Volume Hierarchies.* White Paper, 2001, OPCODE Programmbibliothek, `www.codercorner.com/Opcode.htm`.

Es werden zwei Möglichkeiten vorgestellt, den Speicherbedarf von Hüllvolumen-Bäumen zur Kollisionserkennung zu reduzieren: Den Verzicht auf das Hüllvolumen bei Blättern des Baums, ermöglicht durch Kollisionstests zwischen Primitiven und Hüllvolumen sowie die Kompression des Baums durch diskrete Quantisierung der gespeicherten Positionen. Mit experimentellen Ergebnissen.

[9] James Klosowski, Martin Held, Joseph Mitchell, Henry Sowizral und Karel Zikan: *Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs.* In *IEEE Transactions on Visualization and Computer Graphics*, Band 4 (1), Januar–März 1998, IEEE, Seite 21–36.

Nach einem kurzen Überblick über bestehende Verfahren zur Kollisionserkennung wird ein Algorithmus auf der Basis von k-dops (engl. discrete orientation polytopes, konvexe Polyeder mit Normalen aus einer festen (kleinen) Menge von Vektoren) als Hüllvolumen ausführlich vorgestellt und experimentell analysiert.

[10] James Klosowski: *Efficient Collision Detection for Interactive 3D Graphics and Virtual Environments.* Dissertation, 1998, Graduate School, Stony Brook University, New York.

Enthält u. a. eine Einführung mit Algorithmen und Sätzen zur Kollisionserkennung sowie Kapitel über hierarchische Methoden zur Kollisionserkennung, insbesondere unter der Verwendung von k-dops (Polyeder mit Normalen aus einer festen (kleinen) Menge von Vektoren) als Hüllkörpern sowie zu Verbesserung, Implementierung und experimentellen Analyse dieses Ansatzes.

[11] Eric Larsen, Stefan Gottschalk, Ming Lin und Dinesh Manocha: *Fast Proximity Queries with Swept Sphere Volumes*. Technischer Bericht TR99-018, 1999, Department of Computer Science, University of North Carolina at Chapel Hill.

Der Bericht ist die Basis für den ICRA 2000 Beitrag [12]. Zusätzlich wird auf weitere Hüllvolumina wie entlang eines Geradensegments „geschwenkte“ Kugelvolumina und Kugeln als Spezialfall (PSS, engl. point swept sphere) eingegangen. Enthält einen Vergleich mit anderen Verfahren (unterschiedliche Hüllvolumina und Traversierungstechniken). Mit experimentellen Ergebnissen.

[12] Eric Larsen, Stefan Gottschalk, Ming Lin und Dinesh Manocha: *Fast Distance Queries with Rectangular Swept Sphere Volumes*. In *IEEE International Conference on Robotics and Automation (ICRA 00)*, San Francisco, Kalifornien, USA, 24.–28. April 2000, Seite 3719–3726, IEEE Robotics & Automation Society.

Ein Algorithmus zur Abstandsberechnung und Toleranzprüfung wird vorgestellt, der auf einer hierarchischen Repräsentation mit auf einem Quader „geschwenkten“ Kugelvolumina (RSS, engl. rectangle swept sphere) als Hüllvolumen beruht. Es wird auf einen statistischen Ansatz (Kovarianz-Matrix, top-down) zur Konstruktion des Hüllvolumen-Baums und dessen schnelle Traversierung (prioritätsgerichtete Suche und Ausnutzung von Kohärenz) eingegangen. Mit experimentellen Ergebnissen.

[13] U. Bordon, Marcos Salonia und Heinz Wörn: *Schnelle Kollisionserkennung für mehrere Roboter*. In *VDI-Berichte*, Band 1679, *Robotik 2002*, Ludwigsburg, Deutschland, 19.–20. Juni 2002, VDI, Düsseldorf.

Die Autoren stellen ein System zur Kollisionserkennung bzw. Abstandsberechnung für mehrere Roboter in Echtzeit vor. Die Roboter werden durch vorausberechnete, verschieden genaue Hüllvolumina (Kugeln, achsenausgerichtete Quader, orientierte Quader, konvexe Hüllen) approximiert, wobei nur im Bedarfsfall bis auf die genaue Objektgeometrie zurückgegangen wird. Konkave Objekte werden in konvexe Objekte zerlegt.

GJK-Algorithmus

[14] Stephen Cameron: *Enhancing GJK: Computing Minimum and Penetration Distances Between Convex Polyhedra*. In *International Conference on Robots and Automation (ICRA 97)*, Albuquerque, New Mexico, USA, 20.–25. April 1997, IEEE Robotics & Automation Society.

Stellt einen durch die Hinzunahme eines Gradientenabstiegsverfahren modifizierten GJK-Algorithmus zur Kollisionserkennung vor, der auch die Durchdringungstiefe zweier kollidierender Objekte berechnet. Die zeitliche Komplexität entspricht der des Lin-Canny-Algorithmus. Mit experimentellen Ergebnissen.

[15] Gino van den Bergen: *A Fast and Robust GJK Implementation for Collision Detection of Convex Objects*. In *Journal of Graphics Tools*, Band 4 (2), 1999, ACM, Seite 7–25.

Präsentiert eine Verbesserung des GJK-Algorithmus durch Vorhalten von Daten (bereits berechneten Skalarprodukten), geschicktere Auswahl von Teilsimplizia, frühere Terminierung beim Finden einer separierenden Achse und Ausnutzen zeitlicher Kohärenz. Ein numerisches Terminierungsproblem des ursprünglichen Algorithmus wird behoben. Mit experimentellen Ergebnissen.

Lin-Canny-Algorithmus

[16] Ming Lin und John Canny: *A Fast Algorithm for Incremental Distance Calculation.* In *IEEE International Conference on Robotics and Automation (ICRA 91)*, Sacramento, Kalifornien, USA, 9.–11. April 1991, Seite 1008–1014, IEEE Robotics & Automation Society.

Beschreibt ausführlich den Lin-Canny-Algorithmus zur Kollisionserkennung und Distanzberechnung. Mit experimentellen Ergebnissen.

[17] Brian Mirtich: *V-Clip: Fast and Robust Polyhedral Collision Detection.* In *ACM Transactions on Graphics*, Band 17 (3), Juli 1998, ACM, Seite 177–208.

Beschreibt den Voronoi-Clip-Algorithmus. Dieser verfolgt, ähnlich dem Lin-Canny-Algorithmus, jedoch ohne numerische Toleranzschranken, die einander nächsten Merkmale zweier konvexer Polyeder. Der Fall einander durchdringender Polyeder wird vom Algorithmus korrekt behandelt. Mit experimentellen Ergebnissen.

[18] Stephen Ehmann und Ming Lin: *Accelerated Proximity Queries Between Convex Polyhedra by Multi-Level Voronoi Marching.* In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Takamatsu, Japan, 31. Oktober–5. November 2000, Seite 2101–2106 (3).

Beschreibt einen Algorithmus zur Kollisionserkennung und Entfernungsbestimmung für sich bewegende konvexe Polyeder. Der Algorithmus basiert auf einer hierarchischen Approximation der Objekte, Voronoi-Diagrammen und einer Variante des Lin-Canny-Algorithmus. Mit experimentellen Ergebnissen.

Wegplanung

Überblick und Allgemeines

[19] Micha Sharir: *Algorithmic Motion Planning.* In Jacob Goodman and Joseph O'Rourke (Herausgeber): *Handbook of Discrete and Computational Geometry*, 1997, CRC, Seite 733–754.

Beschreibt von einem formalen geometrisch-algebraischen Standpunkt aus allgemeine Algorithmen und Komplexitätsergebnisse für das Wegplanungsproblem, spezialisierte Algorithmen für wenige Freiheitsgrade und weiterführende Problemstellungen. Enthält einen Abschnitt zu Davenport-Schinzel-Sequenzen sowie eine Liste weiterer Übersichtsartikel zur Wegplanung.

[20] Jean-Claude Latombe: *Motion Planning: A Journey of Robots, Molecules, Digital Actors, and Other Artifacts.* In *International Journal of Robotics Research, Special Issue on Robotics at the Millennium – Part I*, Band 18 (11), November 1999, SAGE, Seite 1119–1128.

Beschreibt bestehende theoretische und praktische Methoden zur Wegplanung und ihre Entwicklung, Ergebnisse zur Komplexität, neuere und verwandte Fragestellungen, Anwendungen sowie mögliche zukünftige Entwicklungen.

[21] Steven LaValle: *Planning Algorithms*, Vorabversion, Stand 31. Juli 2003, `msl.cs.uiuc.edu/planning/`.

Derzeit teilweise fertig gestellter Buchentwurf. Aus Sicht von Robotik, Kontrolltheorie und künstlicher Intelligenz werden u. a. kombinatorische, probabilistische und kontinuierliche Algorithmen für das Pfadplanungsproblem und Varianten davon behandelt. Enthält Kapitel zu geometrischer Repräsentation sowie Entscheidungs- und Spieltheorie.

Stochastische Straßenkarten

[22] Lydia Kavraki und Jean-Claude Latombe: *Probabilistic Roadmaps for Robot Path Planning.* In Kamal Gupta und Angel del Pobil (Herausgeber): *Practical Motion Planning in Robotics: Current Approaches and Future Directions*, 1998, John Wiley & Sons, Seite 33–53.

Das Kapitel beschreibt den Ansatz stochastischer Straßenkarten zur Wegplanung. Beinhaltet eine Einführung in das Thema, eine ausführliche Beschreibung des Algorithmus, einige experimentelle Ergebnisse (Roboter aus linearen Segmenten in zwei Dimensionen mit sechs Freiheitsgraden, aus linearen Segmenten in drei Dimensionen mit sechzehn Freiheitsgraden und ein sich drehender starrer Roboter in drei Dimensionen) und eine theoretische Analyse.

[23] Andrew Ladd und Lydia Kavraki: *Generalizing the Analysis of PRM.* In *IEEE International Conference on Robotics and Automation (ICRA 02)*, Washington DC, USA, 11.–15. Mai 2002, Seite 2120–2125, IEEE Robotics & Automation Society.

Eine Verallgemeinerung der Pfadisolierungsmethode zur Analyse des PRM-Algorithmus wird vorgestellt. Das Problem wird unter Verzicht auf geometrische Einschränkungen als Berechnung der transitiven Hülle einer Relation über einem Wahrscheinlichkeitsraum formuliert. Es wird auf zwei Varianten des Pfadplanungsproblems eingegangen, Roboter mit $2k$ Freiheitsgraden und deformierbare Roboter.

Andere Methoden

[24] John Reif: *Complexity of the Mover's Problem and Generalizations.* In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, San Juan, Puerto Rico, USA, 29.–31. Oktober 1979, Seite 421–427, IEEE.

Der Autor gibt einen polynomiell zeitbeschränkten Algorithmus für ein Wegplanungsproblem mit einem Polyeder als Roboter und Polyedern als Hindernissen. Weiter beweist er durch direkte Reduktion, dass die Problemvariante mit einem Roboter aus mehreren (dreidimensionalen) Polyedern, die durch Gelenke (an Eckpunkten) miteinander verbunden sind, P-SPACE vollständig ist und gibt einen entsprechenden Algorithmus mit polynomiellem Platzbedarf.

[25] Jérôme Barraquand und Jean-Claude Latombe: *Robot Motion Planning: A Distributed Representation Approach.* In *International Journal of Robotics Research*, Band 10 (6), Dezember 1991, SAGE, Seite 628–649.

Stellt einen auf der Potentialfeldmethode basierenden randomisierten Wegplaner vor. Die lokalen Minima des Potentialfelds bilden die Knoten eines Graphen, innerhalb dessen nach dem gewünschten Weg gesucht wird. Zur Ermittlung der lokalen Minima kommen vollständige Enumeration (deterministisch exakter Planer für wenige Freiheitsgrade) und Brownsche Bewegungen (probabilistisch exakter Planer für viele Freiheitsgrade) zum Einsatz. Mit experimentellen Ergebnissen für Roboter mit bis zu 31 Freiheitsgraden.

[26] Mitul Saha, Gildardo Sànchez-Ante und Jean-Claude Latombe: *Planning Multi-Goal Tours for Robot Arms.* In *IEEE International Conference on Robotics and Automation (ICRA 03)*, Taipei, Taiwan, 12.–17. Mai 2003, IEEE Robotics & Automation Society.

Ein gieriger Algorithmus für eine Kombination aus TSP und dem Pfadplanungsproblem für einen Industrieroboter wird vorgestellt. Ziel ist dabei die Einsparung von Pfadberechnungen bei der Lösung des TSP. Mit experimentellen Ergebnissen.

Robotik

Bahnplanung

[27] Sonja Macfarlane und Elizabeth Croft: *Design of Jerk Bounded Trajectories for On-Line Industrial Robot Applications*. In *IEEE International Conference on Robotics and Automation (ICRA 01)*, Seoul, Korea, 21.–26. Mai 2001, Seite 979–984, IEEE Robotics & Automation Society.

Robotertrajektorien, die Erschütterungen bzw. ruckartige Bewegungen vermeiden, verringern die Abnutzung und werden besser nachgeführt (was in der Praxis zu einer höheren Geschwindigkeit führt). Vorgestellt wird eine effiziente Methode zur Berechnung schneller, ruckbegrenzter Punkt-zu-Punkt-Bahnen (engl. linear segments with parabolic bounds) auf der Grundlage von Polynomen fünften Grades.

[28] Björn Hein, Marcos Salonia und Heinz Wörn: *Automated Generated Collision-Free Time Optimized Robot Movements in Industrial Environments Based on Rounding*. In *The 4th International Symposium on Assembly and Task Planning (ISATP 01)*, Fukuoka, Japan, 28.–30. Mai 2001, IEEE Robotics & Automation Society.

Es wird gezeigt, wie man aus einem aus Geradenstücken bestehenden Weg einen kollisionsfreien Weg erstellt, der die in den meisten Roboter-Steuereinheiten vorhandenen Kommandos „fly by“ bzw. „rounding“ verwendet. Die resultierenden Bahnen sind erschütterungsärmer und werden in der Praxis auch schneller ausgeführt.

Roboter

[29] Eberhard Hofer und Klaus Dietmayer: *Praktikum Mess- und Automatisierungstechnik: Programmierung von Industrierobotern*. Praktikumsunterlagen, 2003, Abteilung Mess-, Regel- und Mikrotechnik, Universität Ulm.

Enthält einen kurzen Abschnitt mit grundlegenden Informationen zu Industrierobotern, u. a. zu Aufbau, Klassifikation, Kinematik, Matrizen und Transformationen sowie Bahnplanung und Programmierung.

[30] KUKA Roboter GmbH: *Spezifikation* KR 125 (F)/2, KR 150 (F)/2, KR 200 (F)/2, KR 125 W/2, 25. September 2001, `www.kuka-roboter.de/deutsch/index.html` → Produkte/Applikationen → Spezifikationen.

Gibt einen Überblick über die im Titel aufgeführten Robotermodelle, deren Aufstellung, verfügbares Zubehör und technische Daten wie Abmessungen, Belastbarkeit etc.

[31] KUKA Roboter GmbH: *Spezifikation* KR 150 K, KR 180 K, KR 210 K, 25. September 2001, `www.kuka-roboter.de/deutsch/index.html` → Produkte/Applikationen → Spezifikationen.

Gibt einen Überblick über die im Titel aufgeführten Robotermodelle, deren Aufstellung, verfügbares Zubehör und technische Daten wie Abmessungen, Belastbarkeit etc.

Optimierung

Überblick und Allgemeines

[32] Arnold Neumaier: *Global Optimization*, World Wide Web, 2003, `www.mat.univie.ac.at/~neum/glopt.html`.

Eine umfangreiche Sammlung zum Thema Optimierung, u. a. Methoden (Verzweige & Begrenze-Algorithmen, simuliertes Abkühlen, genetische Algorithmen, Fortsetzungsmethoden), Programmpakete (globale und lokale Optimierung in sechzehn Kategorien), Anwendungen, Testinstanzen und Ergebnisse sowie eine Auflistung von Personen mit Bezug zur Optimierung, ausgewählte Artikel und Hinweise auf Konferenzen, verwandte Themen und andere Internetquellen zum Thema Optimierung.

[33] Perry Gray, William Hart, Laura Painton, Cindy Phillips, Mike Trahan und John Wagner: *A Survey of Global Optimization Methods*, World Wide Web, 2003, `www.cs.sandia.gov/opt/survey/`.

Gibt einen Überblick über Verfahren zur globalen Optimierung, u. a. Verzweige & Begrenze-Algorithmen, Clustering, evolutionäre Algorithmen, hybride Methoden, simuliertes Abkühlen, statistische Methoden und Tabu-Suche. Für jeden Ansatz werden eine Beschreibung, bestehende Anwendungen, Programmpakete und Referenzen angegeben.

[34] Aimo Törn: *Global Optimization*, World Wide Web, 2001, `www.abo.fi/~atorn/ProbAlg/Page510.html`.

Folien zur globalen Optimierung. Enthält eine kurze Einführung, Literaturhinweise, ausgewählte Methoden und Bemerkungen zu Lösbarkeit, Stoppkriterien und Vergleichbarkeit.

[35] Hans Mittelmann und Peter Spellucci: *Decision Tree for Optimization Software*, World Wide Web, 2003, `plato.asu.edu/guide.html`.

Der Schwerpunkt dieses Webangebots zum Thema Optimierung liegt in der Unterstützung bei der Identifikation geeigneter Optimierungsverfahren und gegebenenfalls entsprechender Programme. Mit Informationen u. a. zu (hauptsächlich frei verfügbaren) Programmpaketen, Problemen, Testdaten, Literatur, Hilfsprogrammen und online verfügbaren Optimierungsdiensten.

[36] Torben Hagerup: *Algorithms for NP-Hard Problems*. Skript, Lehrstuhl Komplexität und Algorithmen, Fachbereich Biologie und Informatik, Johann-Wolfgang-Goethe Universität, Frankfurt am Main, 13. Januar 2003.

Enthält Kapitel zu grundlegenden Definitionen, gierigen Algorithmen, Approximationsalgorithmen, Verzweige & Begrenze-Algorithmen, dynamischem Programmieren, polynomiell zeitbeschränkten Approximationsschemata, Parametrisierung, planaren Graphen, linearem Programmieren und Grenzen der Approximierbarkeit. Die Darstellung beinhaltet viele, hauptsächlich kombinatorische Optimierungsprobleme. Mit Übungsaufgaben.

Verzweige & Begrenze-Algorithmen

[37] Jens Clausen: *Branch and Bound Algorithms - Principles and Examples*, Notizen, 12. März 1999,

Erklärt ausführlich Idee, Terminologie und Komponenten (Startlösung, Schranke, Auswahlstrategie, Aufteilung des Suchraums) von Verzweige & Begrenze-Algorithmen anhand der drei Beispiele STSP, Partitionierung von Graphen und quadratisches Zuordnungsproblem. Für die letzten beiden Probleme werden empirische Ergebnisse für parallele Verzweige & Begrenze-Algorithmen vorgestellt. Mit Hinweisen zur Implementierung und Aufgaben zum Thema.

[38] István Hernádvölgyi: *Solving the Sequential Ordering Problem with Automatically Generated Lower Bounds*. Technischer Bericht, 25. August 2003, School of Information Technology & Engineering, University of Ottawa, Kanada.

Stellt Verzweige & Begrenze-Algorithmen für das SOP vor. Die unteren Schranken werden mit Hilfe einer Muster-Datenbank erzeugt, die Abstraktionen des Zustandsraums enthält. Mit experimentellen Ergebnissen.

Mathematische Programmierung

[39] Robert Fourer: *Linear Programming Frequently Asked Questions*, World Wide Web, 2000, `http://www-unix.mcs.anl.gov/otc/Guide/faq/linear-programming-faq.html`.

Gibt einen Überblick über lineares und ganzzahliges Programmieren, nicht kommerzielle und kommerzielle Programmpakete, Sammlungen von Probleminstanzen sowie Literatur (auch Internetquellen) zum Thema. Beantwortet kurz einige ausgewählte Fragen.

[40] Robert Fourer: *Nonlinear Programming Frequently Asked Questions*, World Wide Web, 2000, `http://www-unix.mcs.anl.gov/otc/Guide/faq/nonlinear-programming-faq.html`.

Gibt einen Überblick über nicht lineare Programmierung, nicht kommerzielle und kommerzielle Programmpakete, Sammlungen von Probleminstanzen sowie Literatur (auch Internetquellen) zum Thema.

[41] Alexander Schrijver: *Theory of Linear and Integer Programming*, 1998, Wiley.

Behandelt die Theorie linearer (ganzzahliger) Programmierung. Themen sind u. a. lineare Algebra, Gitter und lineare diophantische Gleichungen, Polyeder (Konzepte, Ergebnisse, Struktur), lineare Ungleichungen, Methoden zur linearen Programmierung (Simplex-Algorithmus, Dualität, Relaxation, Khachiyans Methode, Ellipsoid-Methode, Karmarkars Algorithmus) und zur ganzzahligen linearen Programmierung (Komplexität, total unimodulare Matrizen, Schnittflächen, Verzweige & Begrenze-Algorithmen, Langrange-Relaxation).

Fortsetzungs- und Glättungsmethoden

[42] Kien-Ming Ng: *A Continuation Approach for Solving Nonlinear Optimization Problems with Discrete Variables*. Dissertation, 2002, Department of Management Science and Engineering, Stanford University.

Beschreibt die Lösung nicht linearer ganzzahliger Optimierungsprobleme mit linearen Nebenbedingungen durch einen Glättungsansatz auf Basis der logarithmischen Barrierefunktion; die geglätteten Funktionen werden durch die Newton-Methode und Gradientenabstieg gelöst. Enthält eine Konvergenzanalyse des vorgestellten Algorithmus, eine Untersuchung von linearen Systemen mit großen Elementen auf der Diagonalen, ein Kapitel zur Implementierung des Algorithmus sowie eine vergleichende empirische Analyse des Ansatzes anhand der Probleme binäre quadratische Programmierung mit und ohne Nebenbedingungen, Zuteilung von Frequenzen und einer Anwendung aus der Medizin.

[43] Jorge Moré und Zhijun Wu: *Distance Geometry Optimization for Protein Structures*. In *Journal of Global Optimization*, Band 15 (3), Oktober 1999, Kluwer, Seite 219–234.

Die Autoren beschreiben einen Glättungsansatz für das Distanz-Geometrie-Problem zur Bestimmung der Struktur von Proteinen. Die Glättung erfolgt mittels der Gaußtransformation (Konvolution mit der Dichtefunktion der Normalverteilung), als lokales Optimierungsverfahren kommt ein vmlm (engl. variable metric limited memory) Algorithmus zum Einsatz. Enthält

einen empirischen Vergleich mit einem Multistart-Verfahren auf 100 bzw. 200 Atomen großen Fragmenten eines Proteins.

Simuliertes Abkühlen

[44] Nicholas Metropolis, Arianna Rosenbluth, Marshall Rosenbluth, Augusta Teller und Edward Teller: *Equation of State Calculations by Fast Computing Machines.* In *Journal of Chemical Physics*, Band 21 (6), Juni 1953, American Institute of Physics, Seite 1087–1092.

Der Artikel beschreibt eine Variante der Monte-Carlo-Integration zur Berechnung von Materialeigenschaften (Druck bei gegebenem Volumen und Temperatur), den später nach dem Hauptautor benannten Metropolis-Algorithmus. Empirischer Vergleich für einen Spezialfall (zweidimensionale rigide Kugeln) mit zwei anderen Methoden (Approximation des freien Volumens nach Wood und Virialentwicklung mit fünf Termen).

[45] Griff Bilbro und Wesley Snyder: *Optimization of Functions With Many Minima.* In *IEEE Transactions on Systems, Man, and Cybernetics*, Band 21 (4), Juli 1991, IEEE, Seite 840–849.

Die Autoren stellen eine Variante des simulierten Abkühlens auf Basis von durch k-d-Bäumen repräsentierten Intervallen zur Optimierung kontinuierlicher Funktionen vor und geben einige Anwendungen (Anpassen von Verteilungen, Parameterschätzungen, Bestimmung der Positionierung von Objekten im Raum, Optimierung von Farbmischungen, Spektralanalyse, Neuronale Netze). Weiterhin werden ein interaktives Bedienprogramm für den Algorithmus kurz vorgestellt und die Ergebnisse des Algorithmus auf Testdatensätzen empirisch untersucht.

[46] Peter Salamon, Paolo Sibani und Richard Frost: *Facts, Conjectures and Improvements for Simulated Annealing*, 2002, SIAM.

Dieses von einem Mathematiker, einem Physiker und einem Informatiker geschriebene Buch stellt das simulierte Abkühlen aus dem Blickwinkel der statistischen Mechanik heraus dar. Es wird ausführlich auf den Algorithmus und seine Aspekte, u. a. Zielfunktion, Klassen lokaler Züge, Terminierungskriterien, statistische Informationen und Verlaufspläne für die Abkühlung eingegangen. Mit sechs durchgängigen Beispielen (seismische Dekonvolution, chemische Haufen (engl. chemical clusters), quadratisches Zuordnungsproblem (hier engl. infinite range spin glass problem), bipartite Graphen, TSP, Faltung von Proteinen).

Evolutionäre Algorithmen

[47] Jörg Heitkötter und David Beasley: *The Hitch-Hiker's Guide to Evolutionary Computation (FAQ for comp.ai.genetic)*, World Wide Web, 2001, `surf.de.uu.net/encore/www/`.

Enthält Material zu den verschiedenen Arten evolutionärer Algorithmen, ihren Anwendungen, verfügbaren Programmpaketen sowie Angaben zu Literatur einschließlich Zeitschriften und Konferenzen.

[48] Karsten Weicker: *Evolutionäre Algorithmen*, 2002, Teubner.

Eine Einführung in die evolutionäre Optimierung, die von den biologischen Grundlagen aus über die Bestandteile evolutionärer Algorithmen und ihr Zusammenspiel zu den Standardalgorithmen bis hin zu fortgeschrittenen Themen führt. Es werden sowohl der theoretische Hintergrund als auch praktische Aspekte diskutiert.

[49] Ingo Wegener: *Evolutionäre Algorithmen*. Skript, Lehrstuhl Informatik II, Fachbereich Informatik, Universität Dortmund, 2002.

Beinhaltet eine Einführung (Multistart, simuliertes Abkühlen, evolutionäre Algorithmen, hybride Verfahren), Grenzen (NFL, engl. no free lunch theorem) und Entwurf (Bestandteile, Nebenbedingungen, Struktur der Population, adaptive Verfahren, Implementation) randomisierter Suchverfahren. Weitere Themen sind bekannte Fitnessfunktionen, klassische Analyseansätze (stochastische Prozesse, Verhaltensmaße, Schemata, modellgesteuerte Analyse), $(1 + 1)$EA und Black-Box-Komplexität.

[50] Zbigniew Michalewicz: *Genetic Algorithms + Data Structures = Evolution Programs*, 1994, Springer.

Das aus den drei Teilen genetische Algorithmen (problemorientierte Einführung in Aufbau und Funktionsweise, einige speziellere Themen), numerische Optimierung (binäre und reellwertige Repräsentationen, lokale Suche, Nebenbedingungen, Implementation) und evolutionäre Programme (Transport-Problem, TSP, maschinelles Lernen, diverse Graphprobleme) bestehende Buch führt in das Thema problemangepasster genetischer Algorithmen ein. Die Präsentation orientiert sich durchgängig an Beispielproblemen.

[51] Timothy Starkweather, S. McDaniel, Keith Mathias, Darrell Whitley und Chris Whitley: *A Comparison of Genetic Sequencing Operators*. In *Proceedings of the 4th International Conference on Genetic Algorithms (ICGA 91)*, San Mateo, Kalifornien, USA, 13.–16. Juli 1991, Seite 69–76, Morgan Kaufman.

Sechs Sequenzierungsoperatoren (verbesserte Kantenrekombination, zwei reihenfolgebasierte, eine positionsbasierte und eine zyklusbasierte Rekombination sowie die partiell abgebildete Rekombination (PMX, engl. partially mapped crossover)) werden auf zwei Sequenzierungsproblemen (TSP mit 30 Städten und eine Anwendung im Bereich Produktion/Lieferung) verglichen. Mit experimentellen Ergebnissen.

[52] Tobias Blickle und Lothar Thiele: *A Comparison of Selection Schemes Used in Evolutionary Algorithms*. In *Evolutionary Computation*, Band 4 (4), Mai 1997, MIT, Seite 361–394.

Aus theoretischer Sicht werden die Selektionsmethoden Turnierselektion, abschneidende Selektion (engl. truncation selection) sowie lineare und exponentielle rangbasierte Selektion untersucht. Zentrales Konzept ist dabei die Fitnessverteilung (welche Fitnesswerte werden wie oft von der Population angenommen). Die Charakteristika Selektionsintensität, Selektionsvarianz und Verlust der Diversität werden untersucht. Die ONEMAX-Funktion (alle Positionen eines Bitstrings sind auf den Wert Eins zu setzen) wird als Beispiel verwendet.

[53] Mark Shackleton, Rob Shipman und Marc Ebner: *An Investigation of Redundant Genotype-Phenotype Mappings and Their Role in Evolutionary Search*. In *Congress on Evolutionary Computation 2000: Evolution at Work*, San Diego, Kalifornien, USA, 16.–19. Juli 2000, Seite 493–500, IEEE Neural Networks Council.

Die Autoren untersuchen den Einfluss von Redundanz in Genotyp-Phänotyp-Abbildungen bei evolutionären Optimierungsverfahren. Mehrere Kodierungsfunktionen (Identität, zufällige Abbildung, zwei Abstimmungsvarianten, eine auf zellulären Automaten und eine auf zufälligen booleschen Netzen basierende Abbildung) werden auf erreichbare Phänotypen und den Verlauf der Optimierung hin untersucht. Es zeigt sich, dass Redundanz hilfreich sein kann, aber nicht muss.

[54] Guo Tao und Zbigniew Michalewicz: *Inver-Over Operator for the TSP*. In *Lecture Notes in Computer Science*, Band 1498, *Parallel Problem Solving from Nature V (PPSN V)*, Amsterdam, Holland, 27.–30. September 1998, Seite 803–812, Springer.

Die Autoren stellen einen Inversions-Operator für das TSP vor, bei dem die Trennstellen für die Inversion anhand von anderen Individuen aus der Population bestimmt werden. Darauf aufbauend wird ein einfacher, robuster, evolutionärer Algorithmus entworfen. Einschließlich eines kurzen Überblicks über Operatoren für das TSP. Mit experimentellen Ergebnissen.

[55] Yoshida Yukiko und Adachi Nobue: *A Diploid Genetic Algorithm for Preserving Population Diversity — Pseudo-Meiosis GA*. In *Lecture Notes in Computer Science*, Band 866, *Parallel Problem Solving from Nature III (PPSN III)*, Jerusalem, Israel, 9.–14. Oktober 1994, Seite 36–45, Springer.

Zur Erhaltung der Diversität der Population eines genetischen Algorithmus werden diploide Chromosomensätze verwendet, zusammen mit einem Vorgang, welcher der natürlichen Meiose (Zellteilung zur Reduktion von diploiden auf haploide Chromosomensätze) ähnelt. Empirischer Vergleich mit einem konventionellen genetischen Algorithmus auf einem dynamischen TSP.

[56] Ágoston Eiben, Robert Hinterding und Zbigniew Michalewicz: *Parameter Control in Evolutionary Algorithms*. In *IEEE Transactions on Evolutionary Computation*, Band 3 (2), Juli 1999, IEEE Neural Networks Council, Seite 124–141.

Der Übersichtsartikel gibt einen Überblick über bestehende Formen der Parameterkontrolle für evolutionäre Algorithmen und klassifiziert diese, basierend auf der Art der Kontrolle (deterministisch, adaptiv, selbst-adaptiv) und dem betroffenen Bestandteil (Repräsentation, Fitnessfunktion, Variationsoperatoren, Paarungsselektion, Umweltselektion und Population) des Algorithmus. Auf die Kombination von Kontrollmechanismen wird eingegangen; es werden Vorschläge zur Vereinheitlichung der in der Literatur auftretenden inhomogenen Terminologie gemacht.

Tabu-Suche

[57] Alain Hertz, Eric Taillard und Dominique de Werra: *A Tutorial on Tabu Search*. In *Proceedings of Giornate di Lavoro AIRO 95, Enterprise Systems: Management of Technological and Organizational Changes*, Ancona, Italien, 20.–22. September 1995, Seite 13–24.

Es werden Geschichte, Idee und Funktionsweise (Ablauf, Rolle des Gedächtnisses) der Tabu-Suche dargestellt; auf die Relevanz der Modellierung, die schnelle Berechnung der Zielfunktion sowie auf Intensifikation und Diversifikation wird gesondert eingegangen. Drei Anwendungen, die minimale Knotenfärbbarkeit von Graphen, das Finden maximal unabhängiger Knotenmengen und die Aufstellung von Stundenplänen werden vorgestellt.

[58] Alessandro Agnetis und Roberto Macchiaroli: *Modelling and Optimization of the Assembly Process in a Flexible Cell for Aircraft Panel Manufacturing*. In *International Journal of Production Research*, Band 36 (3), März 1998, Taylor & Francis, Seite 815–836.

Es wird eine Tabu-Suche-Heuristik zur Minimierung der Gesamtbearbeitungszeit einer aus zwei Robotern bestehenden Station beschrieben; der Arbeitsablauf wird als SOP modelliert. Dabei liegt der Schwerpunkt auf den Zeiten für das Auswechseln der Werkzeugköpfe und den Restriktionen bezüglich der Reihenfolge der Arbeitsschritte.

Memetische Algorithmen

[59] Pablo Moscato: *Memetic Algorithms' Home Page*, World Wide Web, 2003, `http://www.densis.fee.unicamp.br/~moscato/memetic_home.html`.

Es finden sich u. a. Hinweise auf Konferenzen, eine umfangreiche Liste von Personen mit Bezug zu memetischen Algorithmen, eine Bibliographie sowie Verweise auf weitere Webangebote, auch zu verwandten Themen.

[60] Nicholas Radcliffe und Patrick Surry: *Formal Memetic Algorithms*. In *Lecture Notes in Computer Science*, Band 865, *Evolutionary Computing: Artificial Intelligence and the simulation of Behaviour (AISB) Workshop*, Leeds, England, 11.–13. April 1994, Seite 1–16, Springer.

Ein formaler, repräsentationsunabhängiger Rahmen für memetische Algorithmen sowie drei repräsentationsunabhängige Operatoren zur Rekombination (binomische minimale Mutation, zufällig zusammenstellende Rekombination und verallgemeinerte n-Punkt-Rekombination) werden vorgestellt, zusammen mit einer Gradientenabstiegsmethode zur Vervollständigung teilweise spezifizierter Lösungen. Empirische Analyse auf dem TSP.

Iterierte lokale Suche

[61] Helena Lourenço, Olivier Martin und Thomas Stützle: *Iterated Local Search*. In Fred Glover und Gary Kochenberger (Herausgeber): *Handbook of Metaheuristics*, 2002, Kluwer, Seite 321–353.

Die iterierte lokale Suche wird zusammen mit ihren Eigenschaften wie Struktur der Methode (Startlösung, lokale Suche, Perturbation, Akzeptanzkriterium) und des Suchraums (Übergang zum Raum lokaler Optima) vorgestellt. Auf die Beispiele TSP, QAP, diverse Probleme der Ablaufsteuerung und -planung, minimale bipartite Graphen, MAX-SAT und eine Variante von Steiner-Bäumen wird genauer eingegangen. Das Verfahren wird mit anderen, auf lokaler Suche bzw. Multistart basierenden Verfahren verglichen.

[62] Keld Helsgaun: *An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic*. In *European Journal of Operational Research*, Band 126 (1), 1. Oktober 2000, Elsevier, Seite 106–130.

Beschreibt die als LKH (Lin-Kernighan-Helsgaun) Algorithmus bekannte Heuristik für das TSP. Geht auf den ursprünglichen Lin-Kernighan-Algorithmus sowie auf Verbesserungen ein, u. a. auf Kandidatenmengen (die zu untersuchenden Nachbarn eines Knoten werden mittels minimaler 1-Spannbäume bestimmt), zulässige Züge, die Bedeutung der anfänglichen Touren, die Repräsentation einer Tour, Distanzberechnungen und die Zeit zum Erkennen eines lokalen Optimums (engl. checkout time). Empirische Analyse auf Instanzen der Probleme STSP, ATSP und minimaler Hamiltonpfad, darunter einige pathologische Fälle.

Grasp

[63] Mauricio Resende und Celso Ribeiro: *Greedy Randomized Adaptive Search Procedures*. In Fred Glover und Gary Kochenberger (Herausgeber): *International Series in Operations Research and Management Science*, Band 57, *Handbook of Metaheuristics*, 2003, Kluwer, Seite 219–249.

Die Grasp-Metaheuristik wird vorgestellt, einschließlich einiger Weiterentwicklungen und Verbesserungen (reaktives Grasp, Perturbation der Kosten, unterschiedliche Verteilungen zur Elementwahl, Einbeziehung der Ergebnisse vorheriger Iterationen). Es wird auf Parallelität sowie auf Anwendungen von Grasp eingegangen. Mit experimentellen Ergebnissen.

[64] Silvio Binato, William Hery, David Loewenstern und Mauricio Resende: *A GRASP for Job Shop Scheduling*. In Pierre Hansen und Celso Ribeiro (Herausgeber): *Operations Research/ Computer Science Interfaces*, Band 15, *Essays and Surveys on Metaheuristics*, Oktober 2001, Kluwer, Seite 59–79.

Ein Grasp-Algorithmus mit Intensivierung und Anwendung des POP (engl. proximate optimality principle) für das JSP wird vorgestellt. Die Bewertungsfunktion basiert auf der geringsten Erhöhung der Gesamtbearbeitungszeit; zur lokalen Suche werden 2-Vertauschungen verwendet. Mit experimentellen Ergebnissen.

Andere Methoden

[65] R. Baker Kearfott: *A Brief Introduction to Global Optimization and a Preview of `INTOPT_90`*. Vortrag, 16. Oktober 1997, Department of Mathematics, Statistics and Computer Science, Marquette University, Milwaukee. `www.mscs.mu.edu/~globsol/talks.html`

Geht kurz auf allgemeine Ansätze in der globalen Optimierung ein; ausführlichere Behandlung der Arbeiten von János Pintér, Donald Jones und Kaj Madsen sowie dem `INTOPT_90` Programmpaket.

[66] Montaz Ali, Aimo Törn und Sami Viitanen: *A Numerical Comparison of Some Modified Controlled Random Search Algorithms*. In *Journal of Global Optimization*, Band 11 (4), Dezember 1997, Kluwer, Seite 377–385.

Die Autoren stellen eine auf quadratischer Approximation und der β-Verteilung basierende Modifikation der kontrollierten zufälligen Suche vor. Diese wird mit vier ursprünglicheren, Simplex-basierten Varianten auf dreizehn Datensätzen empirisch verglichen.

[67] Fred Glover, Manuel Laguna und Rafael Martí: *Scatter Search*. In Ashish Ghosh und Shigeyoshi Tsutsui (Herausgeber): *Advances in Evolutionary Computation: Theory and Applications*, 2003, Springer, Seite 519–537.

Geschichte, Motivation, Funktionsweise und Bestandteile (Initialisierung, lokales Suchverfahren, Aufnahme in die Referenzmenge, Auswahl aus der Referenzmenge, Kombination von Lösungen) der Streuungssuche zur globalen Optimierung werden erläutert und am Beispiel der Minimierung eines geschlossenen Ausdrucks in vier Variablen detailliert veranschaulicht.

[68] Marco Dorigo und Gianni Di Caro: *The Ant Colony Optimization Meta-Heuristic*. In David Corne, Marco Dorigo und Fred Glover (Herausgeber): *New Ideas in Optimization*, 1999, McGraw-Hill, Seite 11–32.

Eine Einführung in Geschichte, Hintergrund und Funktionsweise der ACO (engl. Ant Colony Optimization) Metaheuristik. Behandelt Parallelisierbarkeit, enthält eine Übersicht über bisherige Anwendungen und geht auf das TSP und ein Routingproblem ausführlicher ein.

[69] Jonas Mockus: *Bayesian Heuristic Approach to Global Optimization and Examples*. In *Journal of Global Optimization*, Band 22, Januar 2002, Kluwer, Seite 191–203.

Einführung in Bayes-Ansätze zur globalen Optimierung, u. a. direkter Bayes-Ansatz (DBA, engl. direct bayesian approach) für teuer auszuwertende Funktionen in weniger als 20 Variablen und Bayes-Ansatz zur Optimierung der Parameter von Heuristiken (BHA, engl. bayesian heuristic approach). Als Beispiele werden das Rucksackproblem, die Erstellung von Stundenplänen für Schulen und ein Zwei-Matrix-Spiel gegeben.

[70] Mike Schäfer: *Globale und lokale Optimierungsverfahren für dreidimensionale Anordnungsprobleme*. Dissertation, 2002, Mathematisch-Naturwissenschaftliche Fakultät, Rheinische Friedrich-Wilhelms-Universität Bonn.

Zur möglichst dichten Anordnung von Objekten im Raum werden globale (Anordnung der Objekte relativ zueinander; diskrete Drehwinkel) und lokale (genaue Anordnung der Objekte bei gegebener relativer Anordnung; kontinuierliche Drehwinkel) Optimierungsverfahren sowie Kombinationen betrachtet; Nebenbedingungen werden berücksichtigt. Enthält u. a. Kapitel zu Polyederseparation (Distanzpolyeder, Randspur, GJK-Algorithmus), zur Darstellung räumlicher Drehungen und zu Anwendungen (hauptsächlich im Automobilbereich).

TSP

[71] Gerhard Reinelt: *The Traveling Salesman. Computational Solutions for TSP Applications*. In *Lecture Notes in Computer Science*, Band 840, September 1994, Springer.

Praktisch motivierte Monographie zum TSP, enthält u. a. Abschnitte zu Grundlagen, Voronoi-Diagrammen, Delaunay-Triangulierungen, Kandidaten-Mengen, konstruktiven Heuristiken (nächster Nachbar, Einfügen, Spannbäume, gierige Ersparnis), verbessernden Heuristiken (Einfügen von Knoten und Kanten, 2-opt und 3-opt-Züge, Elimination kreuzender Kanten, Lin-Kernighan), weiteren Heuristiken (raumfüllende Kurven, Streifen, partielle Repräsentation, Dekomposition; simuliertes Abkühlen, evolutionäre Ansätze, Tabu-Suche, neuronale Netzwerke), unteren Schranken sowie Anwendungen. Mit experimentellen Ergebnissen.

[72] Gregory Gutin und Abraham Punnen: *The Traveling Salesman Problem and Its Variations*. In Ding-Zhu Du und Panos Pardalos (Herausgeber): *Combinatorial Optimization*, Band 12, 2002, Kluwer.

Sammlung von Aufsätzen zum TSP, seinen Varianten, seiner Komplexität, Lösungsmethoden (exakt, approximativ, probabilistisch, heuristisch), Analysemethoden (theoretisch und experimentell) und Programmpaketen. Eigene Kapitel zu den Varianten Maximum-TSP, verallgemeinertes TSP, Preise sammelndes TSP und Flaschenhals-TSP; zahlreiche weitere Varianten werden im Rahmen anderer Kapitel behandelt.

[73] David Applegate, Robert Bixby, Vašek Chvátal und William Cook: *Solving TSPs*, World Wide Web, 2004, `www.math.princeton.edu/tsp`.

Umfangreiche Sammlung zum Thema TSP, u. a. mit Informationen zu Geschichte, Lösungsmethoden (insbesondere der Cutting-Plane-Methode), Testinstanzen und Anwendungen sowie dem CONCORDE Programmpaket für das TSP.

[74] Gilbert Laporte: *Modeling and Solving Several Classes of Arc Routing Problems as Traveling Salesman Problems*. In *Computers & Operations Research*, Band 24 (11), November 1997, Elsevier, Seite 1057–1061.

Die Probleme UCPP, DCPP, MCPP, WCPP (engl. undirected / directed / mixed / windy chinese postman problem), URPP, DRPP, MRPP, WRPP (engl. undirected / directed / mixed / windy rural postman problem) und SCP (engl. stacker crane problem) werden über das GTSP auf das TSP reduziert. Mit empirischen Ergebnissen auf mehreren zufälligen und zwei echten Datensätzen.

[75] Roy Jonker und Ton Volgenant: *Transforming Asymmetric Into Symmetric Traveling Salesman Problems*. In *Operations Research Letters*, Band 2 (4), November 1983, Elsevier, Seite 161–163.

Die Autoren transformieren ATSP-Instanzen in STSP-Instanzen, wobei sich die Anzahl der Knoten verdoppelt. Sie gehen kurz auf teilweise asymmetrische Instanzen und das TSP mit m Handlungsreisenden ein. Mit experimentellen Ergebnissen.

Andere Optimierungsprobleme

[76] Eugene Lawler, Jan Lenstra, Alexander Rinnooy Kan und David Shmoys: *Sequencing and Scheduling: Algorithms and Complexity.* In Stephen Graves, Alexander Rinnooy Kan und Paul Zipkin (Herausgeber): *Handbooks in Operations Research and Management Science*, Band 4, *Logistics of Production and Inventory*, 1993, North-Holland, Seite 445–522.

Das Kapitel gibt eine Übersicht über das Gebiet der (deterministischen) Ablaufsteuerung und -planung (engl. sequencing and scheduling). Neben Definitionen und einem Klassifikationsschema für die Probleme des Gebiets werden bekannte Algorithmen einschließlich ihrer Komplexität vorgestellt, u. a. für Probleme auf einer oder mehreren (parallelen) Maschinen sowie für Aufgaben, die auf einer oder mehreren Maschinen ausgeführt werden. Auf ressourcenbeschränkte Projektplanung und stochastische Ablaufplanung wird kurz eingegangen.

[77] Dino Ahr und Gerhard Reinelt: *A Survey of Arc Routing Problems.* Technischer Bericht (Vorabversion), 21. März 2003, Institut für Informatik, Universität Heidelberg.

Gibt eine Übersicht über verschiedene Arten von kantenorientierten Routenplanungsproblemen, u. a. Postmann-Probleme, Zyklen-Überdeckungsprobleme und Eulersche Probleme. Mit Angaben zu Definition, Komplexität, Algorithmen und Literatur.

[78] Christopher Beck, Patrick Prosser und Evgeny Selensky: *On the Reformulation of Vehicle Routing Problems and Scheduling Problems.* In *Lecture Notes in Computer Science*, Band 2371, *Proceedings of the 5th International Symposium on Abstraction, Reformulation and Approximation (SARA 02)*, Alberta, Kanada, 2.–4. August 2002, Seite 282–289, Springer.

Die Autoren transformieren CVRPs in OJSPs und umgekehrt; Algorithmen für die eine Problemklasse werden jeweils auf transformierte Instanzen der anderen Problemklasse angewendet. Eine „Kompression“ zur Vorverarbeitung von VRPs wird vorgestellt. Die Auswirkungen von Transformation und Kompression auf die Performanz der Algorithmen und die Unterschiede zwischen den beiden Problemen werden untersucht. Mit experimentellen Ergebnissen.

Verschiedenes

Grundlagen

[79] Lothar Papula: *Mathematik für Ingenieure und Naturwissenschaftler,* Band 3: *Vektoranalysis, Wahrscheinlichkeitsrechnung, Mathematische Statistik, Fehler- und Ausgleichsrechnung*, 2001, Vieweg.

Der Autor erklärt ausführlich, verständlich und mit zahlreichen Beispielen Themen der angewandten Mathematik, u. a. zur Vektoranalysis (Kurven, Flächen, Skalar- und Vektorfelder, Koordinatensysteme, Kurven- und Oberflächenintegrale), Statistik (Grundlagen, Kenngrößen, spezielle Verteilungen, Schätzfunktionen, Konfidenzintervalle, Parametertests, Korrelation) sowie Fehler- und Ausgleichsrechnung (Fehlerarten, Auswertung von Messreihen, Fehlerfortpflanzung, Regression). Enthält Übungsaufgaben mit Lösungen.

[80] Thomas Cormen, Charles Leiserson, Ronald Rivest und Clifford Stein: *Introduction to Algorithms*, 2001, MIT.

Das Lehrbuch vermittelt aus algorithmischer Sicht Grundwissen der Informatik, u. a. zu Sortierverfahren, Datenstrukturen, Entwurfsmethoden für Algorithmen, Graphalgorithmen und einer Auswahl speziellerer Themen. Die Darstellung geht auf theoretische und praktische Aspekte ein.

[81] R. Baker Kearfott: *Interval Computations: Introduction, Uses, and Resources.* In *Euromath Bulletin*, Band 2 (1), 1996, European Mathematical Trust, Seite 95–112.

Nach einer kurzen Einführung in die Intervall-Arithmetik werden Anwendungen vorgestellt, u. a. aus den Bereichen globale Optimierung, Integralrechnung, Anfangswertprobleme, partielle Differentialgleichungen, Computergrafik, Chemie und Physik. Verweist auf weitere Ressourcen (Literatur, Internet, Programme, Konferenzen).

[82] Albrecht Beutelspacher: *„Das ist o. B. d. A. trivial!“*, 1999, Vieweg.

Humorvoll geschriebene Sammlung von praktischen Tipps zum Formulieren mathematischer Texte. Neben Betrachtungen zum Formulieren an sich gibt es u. a. Abschnitte zur Wahl von Bezeichnern, zu Definitionen, Sätzen, Beweisen u. ä., zur Verwendung einzelner Wörter (wohldefiniert, notwendig, hinreichend, trivial, eineindeutig, kanonisch), zum Einsatz von Zahlen und Symbolen im Text, zum Umgang mit Quantoren, zum Gebrauch von Artikeln sowie dem Konjunktiv, zur Verwendung von „wir“, „man“ und „ich“ sowie dem Lesen mathematischer Texte.

Komplexitätstheorie

[83] Michael Sipser: *Introduction to The Theory of Computation*, 1997, PWS.

Grundlagen der Komplexitätstheorie, behandelt werden u. a. reguläre und kontextfreie Sprachen, Turingmaschinen, Entscheidbarkeit, Reduktionsarten, Komplexitätsklassen (P, NP, PSPACE, L, NL) sowie einige fortgeschrittenere Themen.

[84] Lane Hemaspaandra und Mitsunori Ogihara: *The Complexity Theory Companion*, 2002, Springer.

Forschungsorientiertes (offene Fragen und Literaturhinweise zu jedem Kapitel), nach Techniken (Selbstreduzierbarkeit, Einwegfunktionen, Turnier-Herrsche & Teile, Isolationstechniken, Reduktion durch Zeugen, polynomielle Interpolation, nicht lösbare Gruppen, zufällige Restriktionen, Polynome und Lückenfunktion) geordnetes Buch zur Komplexitätstheorie. Enthält einen Anhang zu Komplexitätsklassen und einen Anhang zu Reduktionsarten.

[85] Mark Krentel: *The Complexity of Optimization Problems.* In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, Berkeley, Kalifornien, USA, 28.–30. Mai 1986, Seite 69–76, ACM.

Die Komplexitätsklasse OptP für Optimierungsprobleme (in ihrer natürlichen Form, nicht als Entscheidungsprobleme) wird eingeführt; diese erlaubt Unterscheidungen hinsichtlich der Komplexität zwischen Problemen wie TSP, Clique und BPP (engl. bin packing problem). Es werden Teilklassen und eine Reduktionsart definiert; OptP wird in Beziehung zu $\mathrm{P}^{\mathrm{SAT}}$ gesetzt und Separationssätze bewiesen.

Weitere Quellen

[86] Vijay Vazirani: *Approximation Algorithms*, 2001, Springer.

Im ersten der drei Teile des Buches werden kombinatorische Algorithmen für zehn bekannte Probleme (darunter kürzeste Superstrings, TSP, Rucksack-Problem, Bin-Packing) vorgestellt, wobei auf die individuellen Besonderheiten der Probleme eingegangen wird. Der zweite Teil behandelt lineares Programmieren (u. a. Dualität, Runden, Schnitte, Relaxation), ebenfalls anhand zahlreicher Beispiele. Im dritten Teil werden die Themen kürzester Vektor eines Gitters, Zählprobleme und Härte der Approximation behandelt sowie offene Probleme gelistet. Enthält Appendizes zu Komplexitätstheorie und grundlegender Wahrscheinlichkeitsrechnung.

[87] *The Welding Processes: Resistance Welding*, Elektronischer Artikel, 2003, `www.key-to-steel.com` → Articles.

Der Artikel zählt sieben Arten des elektrischen Widerstandsschweißens auf und beschreibt diese kurz. Das Webangebot selbst ist ein kostenpflichtiger Informationsdienst für die stahlverarbeitende Industrie; es werden hauptsächlich Informationen (chemische Zusammensetzung, mechanische und temperaturbezogene Eigenschaften, Querverweise zu Standards usw.) zu zahlreichen Stahlarten angeboten.

[88] *Frankfurt Consulting Engineers / Beratende Ingenieure Frankfurt*, Altmünsterstraße 2d, D-65439 Flörsheim. `www.frankfurt-consulting.de`.

Die mittelständische Firma hat sich auf mathematische Optimierung für Kunden aus der Industrie und dem Ingenieurwesen spezialisiert. Schwerpunkte sind kombinatorische Optimierung, Stochastik und Statistik. Sie hat u. a. das auf einem Simulierten Abkühlen Ansatz basierende Programm OptiSpot zur Optimierung von Roboterschweißzellen entwickelt.

[89] *KUKA Roboter GmbH*, Zugspitzstraße 140, D-86165 Augsburg. Postfach 431364, D-86073 Augsburg. `www.kuka-roboter.de`.

Führender Hersteller von Industrierobotern. Muttergesellschaft ist die IWKA AG mit Sitz in Karlsruhe. Entwicklung, Herstellung und Vertrieb von Gelenk- und Portalrobotern, Lineareinheiten, Steuerungen sowie Schulung, Service, Ersatzteilversorgung und Reparatur.