Fachgebiet | Fakultät Maschinenbau | Technische Universität Dortmund
IT in Produktion und Logistik
Univ.-Prof. Dr.-Ing. Markus Rabe

Bachelorarbeit

Vergleich zwischen Algorithmen zur Optimierung logistischer Netzwerke

Bearbeitet von: Robin Cybulski
Studiengang: B.Sc. Logistik
Matrikelnummer: 166990

Ausgegeben am: 12.07.2018
Eingereicht am: 02.10.2018

Prüfer: Univ.-Prof. Dr.-Ing. Markus Rabe
Betreuer: M.Sc. Majsa Ammouriova

Eidesstattliche Versicherung
(Affidavit)

Cybulski, Robin
Name, Vorname
(Last name, first name)

166990
Matrikelnr.
(Enrollment number)

Ich versichere hiermit an Eides statt, dass ich die vorliegende Bachelorarbeit/~~Masterarbeit~~* mit dem folgenden Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present Bachelor's/~~Master's~~* thesis with the following title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution.

Titel der Bachelor-/~~Masterarbeit~~*:
(Title of the Bachelor's/ ~~Master's~~* thesis):

Vergleich zwischen Algorithmen zur Optimierung logistischer Netzwerke

*Nichtzutreffendes bitte streichen
(Please choose the appropriate)

Dortmund, 02.10.2018
Ort, Datum
(Place, date)

R. Cybulski
Unterschrift
(Signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird gfls. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to €50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, section 63, subsection 5 of the North Rhine-Westphalia Higher Education Act (*Hochschulgesetz*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:**

Dortmund, 02.10.2018
Ort, Datum
(Place, date)

R. Cybulski
Unterschrift
(Signature)

**Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.

Inhaltsverzeichnis

Abkürzungsverzeichnis

ACO	Ant Colony Optimization
ACS	Ant Colony System
AS	Ant System
ATSP	Asymmetrisches Traveling Salesman Problem
ATHENE-Projekt	Applied Theories Enabling Network Excellence Projekt
B&B-Verfahren	Branch-and-Bound-Verfahren
BPSO	Binäre Partikelschwarmoptimierung
CVRP	Capacitated Vehicle Routing Problem
DPSO	Diskrete Partikelschwarmoptimierung
EA	Evolutionärer Algorithmus
EP	Evolutionäre Programmierung
ES	Evolutionsstrategien
EVOP	Evolutionary Operation
GA	Genetischer Algorithmus
GP	Genetische Programmierung
HAS	Hybrid Ant System
HICGS	Hybrid Harmony Improvised Consultant Guided Search
HM	Harmony Search Memory
HS	Harmony Search
HSFLA	Hybrid Shuffled Frog Leaping Algorithmus
HVRP	Heterogeneous Vehicle Routing Problem
IACS	Improved Ant Colony System
IPSO	Integer Partikelschwarmoptimierung
MDVRP	Multi Depot Vehicle Routing Problem
MMAS	MAX-MIN Ant System

MPMSFLA	Multi-phase modified Shuffled Frog Leaping Algorithmus
OBCHS	Opposotion-based Chaotic Harmony Search
PGA	Partial Local Search Genetic Algorithm
PICAO	Parallel Improved Ant Colony optimization
PSO	Partikelschwarmoptimierung
PTS	Pheromone Trail Smoothing
QAP	Quadratic Assignment Problem
RHGA	Route-directed hybrid Genetic Algorithm
SFLA	Shuffled Frog Leaping Algorithmus
S-PSO	Set-based Particle Swarm Optimization
STSP	Symmetrisches Traveling Salesman Problem
TSP	Traveling Salesman Problem
UPSO	Unified Particle Swarm Optimization
UWLP	Uncapacitated Warehouse Location Problem
VRPTW	Vehicle Routing Problem with Time Windows
WLP	Warehouse Location Problem

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die Welt der Logistik bietet eine Vielzahl von grundlegend verschiedenen Herausforderungen, welche es möglichst effizient und kostengünstig zu bewältigen gilt. Der Einfluss der logistischen Entscheidungen auf den Erfolg eines Konzerns ist von essenzieller Bedeutung um konkurrenzfähig zu bleiben. Bereits in der Planungsphase eines Unternehmens spielt die Logistik eine tragende Rolle. So kann beispielsweise durch die Wahl eines ungünstigen Standortes die Existenz eines ganzes Betriebes gefährdet werden, noch bevor es sich auf dem Markt etablieren kann. Besonders durch die immer weiter voranschreitende Globalisierung und die damit verbundene Möglichkeit für Unternehmen, ihre Erzeugnisse weltweit zu vermarkten, ist die Bedeutung der Logistik gestiegen, was sich durch den starken Wachstum des Umsatzes des Logistikmarktes zwischen 2000 (153 Mrd. €) und 2015 (240 Mrd. €) in Deutschland belegen lässt (Arnold-Rothmaier 2016, S. 32). Gleichzeitig steigt auch der Stellenwert logistischer Netzwerke. Durch die länger werdenden Transportwege im Zuge der Globalisierung, des immer weiter voranschreitenden Wettbewerbs und der erhöhten Kundenanforderungen ist ein optimaler Ablauf zwischen den Parteien innerhalb der Netzwerke unabdingbar (Gomm und Trumpfheller 2004, S. 45). Durch die Optimierung des Ablaufs können unnötige Kosten und hohe Einsparungspotentiale aufgedeckt werden.

Das Ziel der vorliegenden Arbeit ist, verschiedene Algorithmen und Heuristiken kritisch zu hinterfragen und hinsichtlich ihrer Anwendungsgebiete innerhalb logistischer Netzwerke zu kategorisieren. Dazu wurde bereits im Vorfeld beschlossen sich auf Algorithmen zu fokussieren, dessen grundsätzliche Logik auf Phänomene aus der Natur zurückzuführen ist. So wurde beispielsweise für einige Algorithmen das Verhalten von Tierschwärmen oder Ameisenkolonien analysiert und eine gewisse Regelmäßigkeit festgestellt. Die Logik dahinter wurde mithilfe mathematischer Operatoren in eine oder mehrere Zielfunktionen übertragen, um somit Lösungen für Optimierungsprobleme zu erarbeiten. Im Mittelpunkt dieser Arbeit stehen die Heuristiken Harmony Search Algorithmus, Ameisenalgorithmus, Shuffled Frog Leaping Algorithmus, Partikelschwarmoptimierung sowie der Evolutionäre Algorithmus.

Um das Ziel zu erreichen, ist es nebst den Grundlagen erforderlich, diese Algorithmen in Bezug auf die zu optimierenden Zielfunktionen, ihren Entscheidungsvariablen und ihrer Vorgehensweise miteinander zu vergleichen und ihre Tauglichkeit für individuelle Problemstellungen innerhalb logistischer Netzwerke begründet zu erarbeiten. Desweiteren werden realitätsnahe Benchmarks verwendet und auf Basis der erarbeiteten Erkenntnisse die Güte der Algorithmen zur Lösung güterflussorientierter Probleme überprüft, um als Ergebnis eine strukturierte Übersicht zu erhalten in welchen Bereichen logistischer Netzwerke die beschriebenen Algorithmen zur Optimierung beitragen können und wo ihre Grenzen liegen.

2 Grundlagen zur Optimierung logistischer Netzwerke

In diesem Kapitel werden alle benötigten theoretischen Hintergründe beschrieben, welche zur Optimierung logistischer Netzwerke erforderlich sind. Neben Erläuterungen zur Entstehung und Bedeutung logistischer Netzwerke und der Funktionsweise exakter Algorithmen, stehen vor allem Heuristiken im Fokus dieser Arbeit. Insbesondere wird auf den Ursprung und die Logik naturanaloger Algorithmen eingegangen, welche im weiteren Verlauf der Arbeit im Mittelpunkt stehen.

2.1 Logistische Netzwerke

Unter einem logistischen Netzwerk versteht man "eine auf die Realisierung von Wettbewerbsvorteilen zielende Organisationsform ökonomischer Aktivitäten, die sich durch komplex-reziproke, eher kooperative denn kompetitive und relativ stabile Beziehungen zwischen rechtlich selbstständigen, wirtschaftlich jedoch zumeist abhängigen Unternehmen auszeichnen" (Gomm und Trumpfheller 2004, S. 47), (Sydow 1992, S. 79). Dies bedeutet, dass neben der Konzentration auf die Erreichung interner Unternehmensziele, auch Lösungen für gemeinsame Zielvorgaben der kooperierenden Unternehmen innerhalb eines logistischen Netzwerkes angestrebt werden (Gomm und Trumpfheller 2004, S. 47). In **Abbildung 2-1** ist ein solches Netzwerk vereinfacht dargestellt. Ausgehend vom Startpunkt, der *Quelle*, durchläuft das Objekt entlang der Kanten (Transportwege) mehrere Zwischenknoten in denen es beispielsweise weiterverarbeitet, verpackt oder zwischengelagert wird, bis es letztendlich ihren Endbestimmungsort, auch *Senke* genannt, erreicht. Je nach gefordertem Detaillierungsgrad, kann das Modell auf verschiedenen Abstraktionsniveaus angefertigt werden, sodass die Knoten einzelne Stellplätze, Umschlaglager oder sogar ganze Absatzmärkte darstellen können (Dittmann 2006, S. 19)[1].

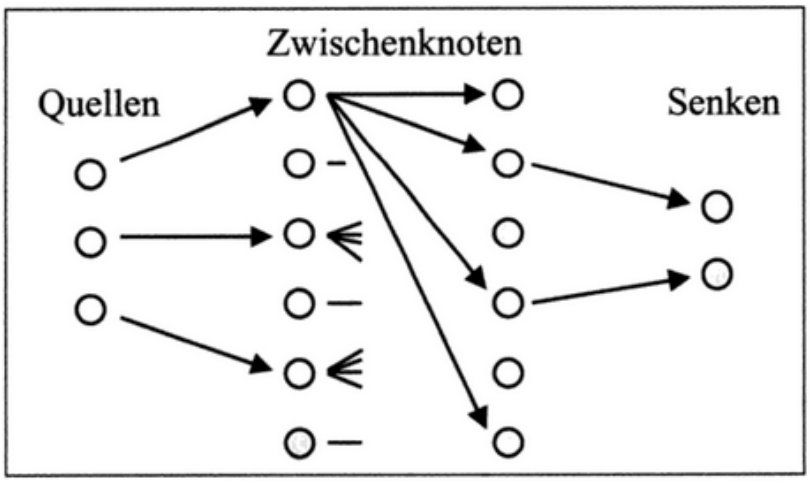


Abbildung 2-1: Einfacher Objektfluss von Quellen zu Senken in einem Netzwerk (Dittmann 2006, S. 18)[2]

[1] Originalquelle: (Pfohl 2004, S. 106)
[2] Originalquelle: (Richard 2000)

Der Wandel von kurzen, einfachen Wertschöpfungsketten zu internationalen Supply Chains, ausgelöst durch die Globalisierung, hatte zur Folge, dass immer mehr Unternehmen ganze Aufgabenbereiche an Zulieferer, Abnehmer oder Dienstleister abtraten. Durch den Fokus der Unternehmen auf ihre Kernkompetenzen sank somit auch die Möglichkeit durch interne Optimierungsvorgänge Kosten einzusparen (Gomm und Trumpfheller 2004, S. 45), (Weber, Bacher und Groll 2002, S. 133). Dadurch rücken Optimierungsmaßnahmen zur Verbesserung der logistischen Netzwerke immer weiter in den Vordergrund um Kosten zu senken. Es wird sogar prognostiziert, dass in Zukunft der Wettbewerb nicht mehr zwischen einzelnen Konzernen ausgetragen wird, sondern Unternehmensnetzwerke entscheidend für den Erfolg eines Unternehmens werden (Gomm und Trumpfheller 2004, S. 45), (Weber, Bacher und Groll 2002, S. 133).

Eine repräsentative Umfrage, durchgeführt vom Fachgebiet *Unternehmensführung und Logistik* der Technischen Universität Darmstadt im Rahmen des ATHENE-Projektes[3] ergab, dass 84,0% aller befragten Unternehmen eine Kostenreduktion als Ziel einer Kooperation in einem logistischen Netzwerk angaben (siehe **Abbildung 2-2**) (Pfohl, Boldt, et al. 2004, S. 155).

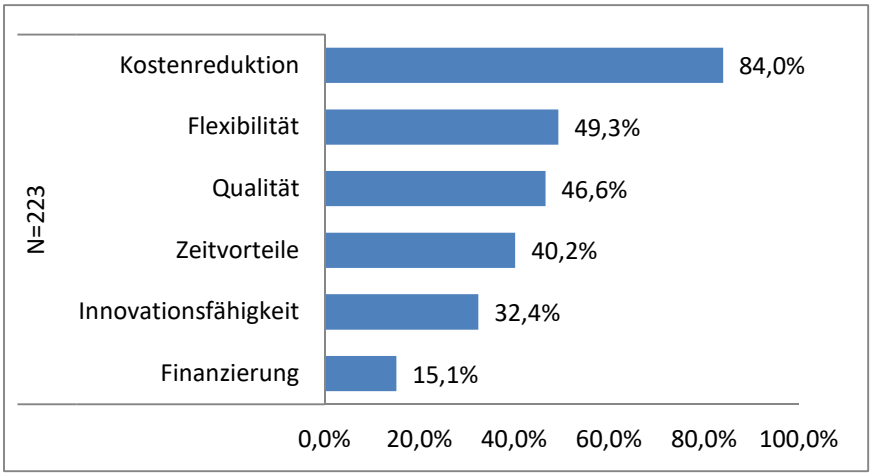


Abbildung 2-2: Teilergebnis der Studie "Erfolgsfaktor Kooperation" (Pfohl, Boldt, et al. 2004, S. 156)

Um das Ziel der Kostenreduktion zu erreichen, steht neben sozialen Aspekten wie beispielsweise gegenseitiges Vertrauen untereinander, Einhaltung von Verträgen oder konkreter Kommunikation vor allem die Optimierung von Güter- und Materialflüssen im Mittelpunkt. Dazu gehören unter anderem eine ideale Routen- und Tourenplanung zur Belieferung der Akteure, die Wahl eines günstigen Standortes um lange Transportwege zu vermeiden, oder auch eine optimale Ausnutzung der zur Verfügung stehenden Lagerfläche. Abhängig von der Anzahl der beteiligten Akteure, benötigten Parametern und Variablen kann eine solche Optimierung eine außerordentlich komplexe Angelegenheit werden, weshalb ausführliche mathematische Berechnungen notwendig sind.

[3] Ausführliche Informationen zum Ablauf und Ergebnisse der Umfrage in (Pfohl, Boldt, et al. 2004, 141-171)

2.2 Exakte Algorithmen

Ein Ansatz um solche mathematischen Optimierungsprobleme lösen zu können, ist die Anwendung exakter Algorithmen. Bei der Verwendung dieser Algorithmen wird stets das globale Optimum, also die bestmögliche Lösung eines Problems, angestrebt. Dies ist aber nur für recht simple Problemstellungen mit wenigen Akteuren und Variablen möglich. Aufgrund der schrittweisen Arbeitsweise von Algorithmen, in denen jede mögliche Lösung berechnet und untersucht wird, steigt der Rechenaufwand und die Laufzeit drastisch, je mehr Akteure involviert sind. Eine häufig herangezogene Benchmark zur Bestimmung der Güte eines Algorithmus zur Lösung von Optimierungsproblemen in logistischen Netzwerken ist das *Traveling Salesman Problem* (TSP). Das TSP bezeichnet die Problemstellung, bei der alle nötigen Orte mittels einer einzigen Tour in einer wegoptimalen Reihenfolge ein Mal besucht werden sollen (Klaus, Krieger und Krupp 2012, S. 593). Unterschieden wird beim TSP in Bezug auf seine Symmetrie. Bei symmetrischen Problemen ist die Distanz d von Knoten i zu Knoten j genauso groß wie von j zu i. Bei asymmetrischen TSP ist dies nicht der Fall und es gilt somit $d_{ij} \neq d_{ji}$. **Tabelle 2-1** zeigt beispielhaft die Anzahl möglicher Lösungen für das symmetrische TSP in Abhängigkeit von der Anzahl der besuchten Städte.

Tabelle 2-1: Anzahl der Lösungen des TSP in Abhängigkeit der besuchten Städte (eigene Darstellung)

Städte	Mögliche Lösungen
3	1
4	3
5	12
6	60
7	360
8	2.520
9	20.160
10	181.440
15	$4{,}36*10^{10}$
20	$6{,}08*10^{16}$
30	$4{,}42*10^{30}$

Anhand dieses typischen Optimierungsproblems logistischer Netzwerke wird die Schwachstelle exakter Algorithmen verdeutlicht. Wird davon ausgegangen, dass *n* die Anzahl der Akteure in einem Netzwerk ist, welcher alle binnen einer Tour beliefert werden müssen, existieren insgesamt $\frac{(n-1)!}{2}$ verschiedene Möglichkeiten in welcher Reihenfolge dies geschehen kann (Grötschel 2007, S. 110). Bereits ab einer Anzahl von $n=15$ beteiligten Akteuren beträgt die Laufzeit 1,3 Jahre, bei $n=16$ sogar 20 Jahre (Näher 2006). Der starke Anstieg der Laufzeit ist durch die fakultäre Abhängigkeit der möglichen Lösungen zu der Anzahl der Akteure zu erklären. Dieses Problem gehört zu der Kategorie der NP-vollständigen Probleme. Diese Art von Problemen ist NP-schwer und liegt selbst in NP was bedeutet, dass kein exakter Algorithmus existiert, welcher das Problem in polynomieller und somit annehmbarer Zeit lösen kann (Gerdes, Klawonn und Kruse 2004, S. 228-229). Da

praktische Problemstellungen in der Regel solche komplexe Strukturen aufweisen, ist der Einsatz exakter Algorithmen aufgrund ihrer ineffizienten Laufzeit ungeeignet und wird deshalb im Verlaufe dieser Arbeit nicht weiter betrachtet.

2.3 Heuristiken

2.3.1 Anwendung von Heuristiken in Abgrenzung zu exakten Algorithmen

Da exakte Algorithmen für realitätsnahe Optimierungsprobleme aus den zuvor genannten Gründen ungeeignet sind, werden Heuristiken zur Lösungsermittlung herangezogen. Diese suchen nicht nach dem globalem Optimum, sondern ermitteln eine gute zulässige Lösung, welche, je nach Güte der Heuristik, nur geringfügig vom Optimum abweicht. Da das globale Optimum oftmals jedoch unbekannt ist, ist nicht ersichtlich wie groß die Abweichung tatsächlich ist (Domschke und Scholl 2010, S. 31). Diese (unbekannte) Abweichung wird jedoch in Kauf genommen, da sich der Rechenaufwand im Vergleich zu exakten Algorithmen drastisch reduziert. Aufgrund der kürzeren Laufzeit, infolge des geringeren Rechenaufwandes, sind Heuristiken der einzig praxistaugliche Lösungsansatz für Optimierungsprobleme (Klaus, Krieger und Krupp 2012, S. 225).

2.3.2 Klassifizierung von Heuristiken

Aufgrund der vielfältigen Einsatzgebiete von Heuristiken existieren sowohl problemspezifische, als auch problemunabhängige Heuristiken, auch Metaheuristiken genannt. Im Falle der problemspezifischen Heuristiken, werden die Lösungsansätze speziell auf ein vorhandenes Problem, wie beispielweise das bereits angesprochene Traveling Salesman Problem oder Tourenplanungen zugeschnitten und eignen sich deshalb auch nur für dieses eine Problem. Metaheuristiken dagegen fokussieren sich nicht auf ein bestimmtes Problem, sondern basieren auf unterschiedlichen allgemeinen Vorgehensprinzipien. Um vorgegebene Probleme mittels Metaheuristiken lösen zu können, sind nur wenige Anpassungen des Modells vonnöten. (Domschke und Scholl 2010, S. 21). Weil der Fokus dieser Arbeit auf den naturbasierten Algorithmen liegt, und diese der Gruppe der Metaheuristiken angehören, wird auf die problemspezifischen Algorithmen nicht genauer eingegangen.

Heuristiken lassen sich je nach ihrem Anwendungszweck in

1) Eröffnungsverfahren
2) Lokale Suchverfahren / Verbesserungsverfahren
3) Populationsverfahren
4) Unvollständig ausgeführte bzw. vorzeitig abgebrochene exakte Verfahren
5) Relaxationsbasierte Verfahren

unterteilen (Domschke und Scholl 2010, S. 21), wobei Populationsverfahren aufgrund der hervorgehobenen Relevanz naturbasierter Algorithmen für diese Arbeit gesondert und ausführlich in Kapitel 3 erläutert werden.

Eröffnungsverfahren ermitteln eine erste zulässige Lösung für ein vorliegendes Problem. Wie weit diese Eröffnungslösung vom Optimum entfernt liegt, ist von dem investierten Rechenaufwand abhängig. Möchte man für den Anfang nur eine zulässige Lösung innerhalb von kurzer Zeit erhalten, ganz egal wie wirtschaftlich sie ist, kann dies mithilfe von nur wenig Rechenaufwand erreicht werden. Strebt man jedoch bereits im Eröffnungsverfahren eine nahezu optimale Lösung an, wird entschieden mehr Rechenkapazität und dementsprechend mehr Zeit benötigt (Domschke und Scholl 2010, S. 21). Auch hierbei erfolgt eine Einteilung der Verfahren. Wird in jedem Iterationsschritt des Eröffnungsverfahrens die größtmögliche Verbesserung zum vorherigen Ergebnis angestrebt, spricht man von einer *Greedy-Heuristik*. Diese Art von Eröffnungsverfahren ist der in der Praxis am häufigsten verwendete Ansatz für das Erstellen einer Ausgangslösung. Ein sehr häufig aufkommendes Beispiel für Greedy-Heuristiken als Eröffnungsverfahren sind die Add- und Drop-Verfahren [4] für Standortprobleme. Eine weitere Art von Eröffnungsverfahren ist das *vorausschauende Verfahren*, in dem die Auswirkungen eines jeden Iterationsschrittes, z.B. durch eine Variablenfixierung, auf den nächsten Schritt abgeschätzt wird (Domschke, Drexl, et al. 2015, S. 136). *Uninformierte Verfahren* hingegen, beschränken sich lediglich darauf eine zulässige Lösung zu finden ohne die Zielfunktion zu berücksichtigen (Domschke, Drexl, et al. 2015, S. 136). Beide Eröffnungsverfahren werden in der Praxis eher selten verwendet.

Lokale Suchverfahren und **Verbesserungsverfahren** benötigen eine Ausgangslösung, auf dessen Basis sie die Zielfunktionswerte der Startlösung durch Iterationsschritte verbessern können (Domschke und Scholl 2010, S. 23). Eine solche Ausgangslösung kann entweder zufällig ausgewählt werden, oder wie in der Praxis üblich, durch ein Eröffnungsverfahren bestimmt werden (Domschke und Scholl 2010, S. 21 und 23). In jeder Iteration wird die momentane Lösung x mit einer Lösung aus der Nachbarschaft $NB(x)$ verglichen, wobei $NB(x)$ alle möglichen Lösungen, welche aus einer Transformation hervorgehen können, enthält. Insbesondere wird zwischen

1) Einer Veränderung von einer Lösung an genau einer Stelle, z.B. das "Kippen eines Bits" (Zustandsänderung eines Bits- von 0 auf 1 oder umgekehrt)
2) Dem Vertauschen von Elementen
3) Und dem Verschieben von Elementen

als Transformationsvorschriften unterschieden (Domschke, Drexl, et al. 2015, S. 136). Geht durch eine Transformation eine Lösung $x' \in NB(x)$ aus x hervor, spricht man auch von einem

[4] Genauere Informationen zu diesen Verfahren sind zu finden in (Arnold, et al. 2007, 98).

Zug, weshalb dieses Verfahren auch *zugbasiertes Verfahren* genannt wird (Domschke und Scholl 2010, S. 23).

Reine Verbesserungsverfahren suchen in jedem Iterationsschritt nur die Lösungen aus, welche auch zu einer Verbesserung des Zielfunktionswertes führen. Existiert in der Umgebung $\mathrm{NB}(x)$ keine Verbesserungsmöglichkeit mehr, ist dies das Abbruchskriterium für das Verfahren, weshalb das endgültige Ergebnis nur ein lokales Optimum beinhaltet, welches unter Umständen deutlich schlechter als das globale Optimum sein kann. Lokale Suchverfahren hingegen lassen eine vorübergehende Verschlechterung des Zielfunktionswertes zu, weshalb es möglich ist lokale Optima wieder zu verlassen (Domschke und Scholl 2010, S. 24). **Abbildung 2-3** veranschaulicht den Vorteil lokaler Suchverfahren gegenüber reinen Verbesserungsverfahren. Ist das reine Verbesserungsverfahren nach einer gewissen Anzahl von Iterationen beim lokalen Optimum angekommen, bricht das Verfahren an dieser Stelle ab und überprüft die weiteren Lösungsmöglichkeiten nicht weiter. Ein lokales Suchverfahren kann durch eine momentane Verschlechterung der Zielfunktion über x^1 zu dem besseren lokalem Optimum x^2 gelangen. Ebenso besteht die Möglichkeit durch weitere Iterationen und zwischenzeitlichen Verschlechterungen x^3 und x^4 das globale Optimum x^* zu erreichen.

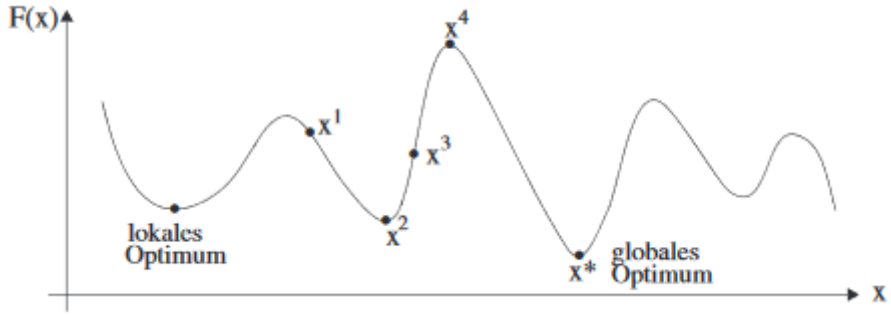


Abbildung 2-3: Darstellung eines möglichen Lösungsverlaufes lokaler Suchverfahren (Domschke und Scholl 2010, S. 25)

Allgemein lassen sich alle bisher genannten Verfahren in deterministische und stochastische bzw. randomisierte Heuristiken einteilen. Deterministische Heuristiken oder Algorithmen liefern bei gleichen Rahmenbedingungen und gleicher Vorgehensweise stets dasselbe Ergebnis, wohingegen es bei stochastischen Verfahren zu unterschiedlichen Ergebnissen bei mehrfacher Anwendung kommen kann (Domschke, Drexl, et al. 2015, S. 137).

Zu den **unvollständig ausgeführten exakten Verfahren** gehört unter anderem das vorzeitig abgebrochene Branch-and-Bound-Verfahren[5]. Ein vorzeitiges Abbruchkriterium kann z.B. eine Restriktion der Rechenzeit oder eine maximale Anzahl an Iterationen sein. Desweiteren besteht die Möglichkeit die Zweige, dessen Auswahl zu einer Verschlechterung um einen gewissen Prozentsatz gegenüber der momentan besten Lösung führen zu ignorieren. Diese Methode besitzt den großen Vorteil, dass durch Vergleich der aktuellen oberen Schranke

[5] Eine genaue Beschreibung des verfahren ist zu finden in (Klaus, Krieger und Krupp 2012, 87).

und der unteren Schranke aller noch nicht eliminierten Teilprobleme, die maximale Abweichung der momentanen Lösung zum globalem Optimum berechnet werden kann (Domschke und Scholl 2010, S. 36-37).

Werden die Nebenbedingungen und Restriktionen eines vorliegenden Problems gelockert und somit vereinfach, spricht man von **relaxationsbasierten Verfahren** (Acker 2011, S. 114). Beispiele solcher Relaxationen sind das *Weglassen von Nebenbedingungen* oder die *LP-Relaxation*, in der die Ganzzahligkeit als Bedingung weggelassen wird, wodurch auch nichtganzzahlige Lösungen zugelassen werden (Domschke und Scholl 2010, S. 37). Fernerhin ist auch die Lagrange-Relaxation ein probates Mittel um eine komplexe Problemstellung zu vereinfachen. Hierbei werden die Nebenbedingungen nicht einfach nur weggelassen, sondern ihre Verletzung in der Zielfunktion berücksichtigt (Klaus, Krieger und Krupp 2012, S. 311).

Desweiteren lassen sich Heuristiken im allgemeinen nach ihren Entscheidungsvariablen unterscheiden (Feldmann 1999, S. 13). Diskrete Variablen können nur endlich viele, oder abzählbar unendliche Ausprägungen annehmen, wie beispielsweise die Augenzahl eines Würfels (Auer, et al. 2013, S. 26). Bei kontinuierlichen oder stetigen Variablen existieren unendlich viele Ausprägungen (Auer, et al. 2013, S. 127). Ein Beispiel hierfür ist die Messung der Zeit oder einer Strecke, da sie theoretisch unendlich genau messbar sind.

2.4 Problemstellungen in logistischen Netzwerken

Neben dem bereits angesprochenem *Traveling Salesman Problem*, dessen Lösung eine wegoptimierte Reihenfolge für Kundenbesuche liefert, existieren noch andere Herausforderungen in logistischen Netzwerken, welche im Folgenden vorgestellt werden. Da die in dieser Arbeit betrachteten Problemstellungen in der Realität hauptsächlich von diskreter Natur sind, werden kontinuierliche Probleme in dieser Ausarbeitung nicht genauer untersucht.

2.4.1 Tourenplanungen

Eine der grundlegendsten Aufgaben der Logistik ist der Transport von Gütern. Um diesen Gütertransport durch eine Transportwegoptimierung möglichst zeit- und kostenminimal und gleichzeitig noch umweltfreundlicher (durch Einsparung von CO_2-Emissionen) gestalten zu können sind Tourenplanungen erforderlich. Dabei ist es unerheblich, ob man den Transport von Gütern zu Kunden, oder zwischen Akteuren innerhalb eines logistischen Netzwerkes betrachtet. Eine kostenminimale Tourenplanung ist für ein optimales Netzwerk unerlässlich. Im Folgenden werden die für diese Ausarbeitung relevanten Tourenplanungen beschrieben sowie ihre Zielfunktionen erläutert. Die jeweils dazugehörigen Nebenbedingungen können im *Anhang A* nachgeschlagen werden.

Traveling Salesman Problem

Obwohl das in Abschnitt 2.2 beschriebene TSP zu den NP-vollständigen Problemen gehört, ist es aufgrund seiner wenigen Nebenbedingungen dennoch das "trivialste" betrachtete Problem in dieser Arbeit. Den einzigen entscheidungsrelevanten Parameter stellt die Minimierung der Strecke bzw. der Kosten dar. Gesucht ist also nach einer Reihenfolge in der jede Stadt genau einmal besucht wird, welche die kürzeste Tour liefert. Daher handelt es sich um ein Minimierungsproblem, welches folgende Zielfunktion aufweist (Domschke und Scholl 2010, S. 99):

$$Min\ F(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \text{ mit} \qquad (2.1)$$

$$x_{ij} = \begin{cases} 1 & falls\ Tour\ Verbindung\ von\ Knoten\ i\ zu\ j\ enthält \\ 0 & sonst \end{cases}$$

Diese Zielfunktion gilt für die Minimierung der Strecke. Sind hingegen die Kosten einzelner Transportwege bekannt und sollen minimiert werden, so wird die Distanzvariable d_{ij} durch eine Kostenvariable c_{ij} ersetz. Desweiteren ist x_{ij} eine Entscheidungsvariable, die sicherstellt, dass nur die Strecken bzw. Kosten in Betracht gezogen werden, welche auch Bestandteil der Tour sind. n ist hierbei die Anzahl der Knoten einer Tour und ist bei der praktischen Lösung des Problems bekannt. Da die Anzahl der Städte abzählbar und endlich ist, handelt es sich hierbei um ein *diskretes* Optimierungsproblem.

Vehicle Routing Problem

Das bisher betrachtete TSP eignet sich eher zur Streckenoptimierung für Handelsvertreter, welche ihre Kunden nur besuchen, aber nicht beliefern müssen. Für eine wegoptimale Belieferung mit Gütern ist es nur bedingt geeignet, da sich das TSP nur nach dem optimalenWeg richtet und Restriktionen, welche in der Realität beim Transport eintreten ignoriert. Somit ist eine Weiterentwicklung dieses Modells nötig. Eine Erweiterung des bisherigem TSP, welche von einer unendlichen Kapazität der LKWs ausgeht, ist das Problem der *kapazitierten Tourenplanung* (Capacitated Vehicle Routing Problem, CVRP). Hierbei wird die begrenzte Ladekapazität der LKWs nicht ignoriert und mittels einer mathematischen Restriktion in die Berechnungen mit einbezogen (Rieck 2008, S. 11). Größere Unternehmen besitzen in der Regel einen Fuhrpark mit unterschiedlichen LKWs, sodass nicht alle Fahrzeuge dieselbe Kapazität aufweisen. Daher müssen die unterschiedlichen Ladevolumina der LKWs durch weitere Nebenbedingungen ebenfalls berücksichtigt werden. Hierbei spricht man von einer *Tourenplanung mit heterogenen Fahrzeugen* (HVRP) (Rieck 2008, S. 19).

Eine weitere Anforderung, welche der Zulieferer unter Umständen erfüllen muss, ist die Einhaltung von Zeitfenstern. Diese Zeiten können von Öffnungszeiten, befristeten Haltebrechtigungen oder organisatorischen Arbeitsabläufen des Kunden (z.B. just-in-time) abhängig sein (Rieck 2008, S. 17). Somit ist der Faktor Zeit eine entscheidungsrelevante Größe, welche in zusätzlichen Nebenbedingungen berücksichtigt werden muss. Dieses

Problem bezeichnet man als *Tourenplanung mit Zeitfenstern* (VRPTW). Allgemein wird bei diesem Problem zwischen zwei Formen unterschieden. Beim originären VRPTW müssen die Zeitfenster strikt eingehalten werden, sodass Fahrzeuge bei einer verfrühten Ankunft warten müssen, bis das Zeitfenster geöffnet ist. Bei weichen VRPTW ist eine Lieferung auch außerhalb des vereinbarten Zeitfensters möglich, was allerdings Strafkosten zur Folge hat (Rieck 2008, S. 17). Auch eine Kombination unterschiedlicher Vehicle Routing Probleme ist möglich, sodass beispielsweise eine Tourenplanung mit heterogenen Fahrzeugen zusätzlich Zeitfenster enthalten kann.

Übersteigt die angeforderte Bedarfsmenge der Kunden die Kapazität der Fahrzeuge, kann die *Tourenplanung mit mehrfachen Fahrzeugeinsatz* (VRPMU) einsetzt werden, vorausgesetzt die Distanzen zwischen den Kunden sind gering (Rieck 2008, S. 23). Dadurch, dass die Fahrzeuge mehrere Touren innerhalb einer Planungsperiode fahren, können andere Fahrzeuge im Depot verbleiben, was langfristig zu einer Verkleinerung des Fuhrparks führen kann (Rieck 2008, S. 23). Erfolgt die Belieferung von mehr als einem Depot aus, so handelt es sich um ein Multi-Depot Vehicle Routing Problem (MDVRP) (Surekha und Sumathi 2011, S. 119). Desweiteren existieren noch andere Tourenplanungen wie z.B. die Tourenplanung mit Auslieferung und Einsammlung, Tourenplanung mit Rücktransport und gemischter Ladung, Tourenplanung mit simultaner Auslieferung und Einsammlung oder das offene Tourenplanungsproblem, welche aber aufgrund ihrer geringen Bedeutung für diese Arbeit nicht genauer beschrieben werden. Weitere Informationen über diese Probleme sind (Rieck 2008, S. 26-30) zu finden. Von Relevanz für den späteren Vergleich sind das CVRP, VRPTW und das MDVRP. Aus diesem Grund werden ihre Zielfunktionen nun genauer untersucht.

Wie bereits eingangs erwähnt, unterscheidet sich das CVRP vom TSP allein durch die Tatsache, dass mehr Nebenbedingungen berücksichtigt werden müssen. Das grundsätzliche Ziel im Zuge einer Kosten- bzw. Streckenminimierung eine optimale Route zur Belieferung der Kunden zu berechnen bleibt gleich. Aus diesem Grund ist die Zielfunktion mit der Formel (2.1) identisch (Rieck 2008, S. 13).

Beim VRPTW dagegen, müssen zwei Zielfunktionen minimiert werden. Als Primärfunktion und somit wichtigeres Ziel, gilt es die Anzahl der gefahrenen Touren bzw. der Fahrzeuge zu minimieren. Da jedes Fahrzeug laut Definition nur eine Tour fährt, ist die Anzahl der Fahrzeuge identisch mit der Anzahl der Touren. Das sekundäre Ziel beinhaltet eine Minimierung der Transportwege. Es sei angemerkt, dass stets eine Lösung mit geringerer Fahrzeuganzahl einer Lösung mit mehr Fahrzeugen vorgezogen wird, auch wenn dies eine längere Fahrzeit mit sich bringt (Gambardella, Taillard und Agazzi 1999, S. 2-3). Die Zielfunktionen für dieses Problem lauten somit (Gong, et al. 2012, S. 256):

$$min\ F_1 = v \tag{2.2}$$

$$\min F_2 = \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{v} d_{ij} * x_{ij}^{k} \tag{2.3}$$

Die Zielfunktion (2.2) beschreibt dabei die Minimierung der Anzahl an genutzten Fahrzeugen v. (2.3) steht für die Minimierung der Distanzen unter Berücksichtigung von v. Wie bei den Problemen zuvor, ist x_{ij}^{k} eine Entscheidungsvariable, welche den Wert 1 annimmt, wenn das Fahrzeug k direkt von Knoten i zu Knoten j fährt.

Bei einer Belieferung von mehreren Depots aus, muss beim MDVRP die Anzahl und die Zuordnung der Depots zu den Kunden in der Zielfunktion berücksichtigt werden. Mathematisch lässt sich dies wie folgt darstellen (Surekha und Sumathi 2011, S. 120):

$$\min F = \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} d_{ij} * x_{ijk} \qquad (2.4)$$

I ist dabei die Menge aller Depots, J die Menge aller Kunden und K die Menge aller Fahrzeuge. Für den Fall, dass das Depot i den Standort j auf der Route k beliefert, nimmt die Entscheidungsvariable x_{ijk} den Wert 1 an und beträgt ansonsten 0.

Bei allen beschriebenen VRP ist die abzählbare, endliche Menge der Kunden sowie ihr Standort bekannt, weshalb auch diese Probleme den diskreten Optimierungsproblemen zuzuordnen sind.

2.4.2 Standortplanungen

Eine gute Tourenplanung ist nicht die einzige Möglichkeit um den Güterfluss in logistischen Netzwerken zu optimieren. Die Wahl eines günstigen Standortes für ein Unternehmen oder ein Lager ist ebenso elementar wie der Transport an sich. Ziel von Standortplanungen ist es, eine geeignete Wahl für einen Standort zu treffen, von dem man die Kunden oder Akteure eines Netzwerkes möglichst weg- und kostenoptimal beliefern kann.

Warehouse Location Problem

Allgemein lassen sich Standortprobleme in volkswirtschaftliche, betriebliche und innerbetriebliche Probleme einteilen (Arnold, et al. 2007, S. 95). Für diese Ausarbeit sind jedoch nur betriebliche Standortplanungsprobleme von Interesse, weshalb die anderen beiden Kategorien nicht genauer erläutert werden. Betriebliche Standortplanungen beschäftigen sich mit der optimalen Standortwahl von Betrieben, verschiedenen Lagern, Umschlagsstationen und auch öffentlichen Einrichtungen wie Krankenhäusern oder Schulen (Arnold, et al. 2007, S. 95). Ist die Nähe zum Kunden, Lieferanten und allen weiteren Akteuren des Netzwerkes ausschlaggebend, so spricht man vom Warehouse Location Problem (WLP) (Arnold, et al. 2007, S. 96). Das WLP geht davon aus, dass die Nachfragemenge der Kunden bekannt ist und potentielle Standorte bereits in einem Vorverfahren lokalisiert wurden, wobei die Anzahl der möglichen Standorte stark variieren kann. Auf dieser Basis werden anschließend die zu eröffnenden Standorte berechnet (Arnold, et al. 2007, S. 97). Unterliegen die Standorte in Bezug auf ihre Lagerkapazität keinen Restriktionen, spricht man von einem *unkapazitiertem WLP*. Existiert eine Grenze in Form einer maximalen Lagerkapazität, wird dies in einer Nebenbedingung berücksichtigt und man bezeichnet dieses Problem als *kapazitiertes WLP* (Arnold, et al. 2007, S. 98).

Zudem wird unterschieden wie viele Transportstufen berücksichtigt werden müssen. Bei *einstufigen WLP* durchläuft der Güterfluss nur eine Transportstufe vom Lager zum Kunden. Sind hingegen mehrere Akteure in der Supply Chain involviert, wodurch mindestens zwei Transportstufen entstehen, spricht man von einem *mehrstufigen WLP* (Arnold, et al. 2007, S. 98-99).

Auch hier ist eine Kombination aus kapazitäts- und stufenbasierten WLP möglich. **Abbildung 2-4** zeigt beispielhaft wie ein Modell des einstufig-kapazitierten-WLP, in der jeder potentielle Standort mit der Kapazitätsrestriktion a_i den Kunden über nur eine Transportstufe direkt beliefert.

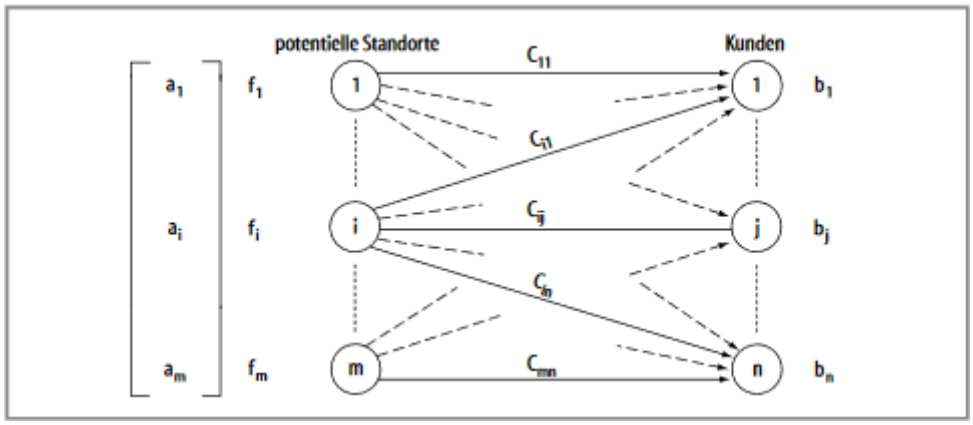


Abbildung 2-4: Struktur eines einstufig-kapazitierten-WLP (Arnold, et al. 2007, S. 97)

Von besonderen Interesse für diese Arbeit ist jedoch das einstufig-unkapazitierte WLP, sodass die Kapazitätsrestriktion a_i wegfällt. Die Zielfunktion dieses Problems lautet wie folgt (Arnold, et al. 2007, S. 97-99):

$$\min F(x,y) = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ji} + \sum_{i=1}^{m} f_i y_i \qquad (2.5)$$

$$\text{mit } y_i \begin{cases} 1 & wenn\ am\ Standort\ i\ ein\ Lager\ errichtet\ wird \\ 0 & sonst \end{cases}$$

Die erste Summe dieser Funktion beschreibt die Transportkosten von m potentiellen Standorten zu n Kunden. Die zweite Summe stellt die Errichtungskosten eines Standortes dar. Dabei ist f_i die Höhe der Kosten und y_i eine binäre Variable, welche bei Standorteröffnung den Wert 1 annimmt und sonst 0 beträgt. x_{ji} ist ebenfalls eine binäre Variable, welche den Wert 1 annimmt wenn der Kunde *i* vom Standort *j* beliefert wird. Da das WLP davon ausgeht, dass im Vorfeld eine Auswahl von potentiellen Standorten getroffen wurde, ist die Anzahl und Lage der Standorte bekannt, was auch dieses Problem zu einem diskreten Optimierungsproblem macht.

Quadratic Assignment Problem

Ein weiteres Problem der Standortplanung ist das Quadratic Assignment Problem (QAP). Hierbei steht nicht die Wegoptimierung zum Kunden, sondern die Strecke zwischen den Betrieben untereinander im Vordergrund. Das QAP hat zum Ziel, einer vorgegebenen Anzahl von Betrieben die optimalen Standorte im Sinne einer Kostenminimierung zuzuordnen. Dabei wird davon ausgegangen, dass bereits in einer Vorauswahl potentielle Standorte lokalisiert wurden und somit aus einer bekannten Menge von Standorten gewählt werden kann. Desweiteren entstehen bei jeder Standorteröffnung zusätzlich Errichtungskosten. Geht man von einer Anzahl von *n* Betrieben aus, liegen die Entscheidungsrelevanten Parameter in *n x n* Matrizen vor. $F = (f_{ij})$ ist die Güterflussmatrix, in der f_{ij} den Fluss von Betrieb i zu Betrieb j darstellt. Durch die Matrix $D = (d_{kl})$ werden die Distanzen zwischen den potentiellen Standorten beschrieben. d_{kl} ist hierbei die Distanz von Standort k zu Standort l. Die Errichtungskosten sind in der Matrix $B = (b_{ik})$ dargestellt, wobei b_{ik} die Kosten des Betriebes i am Standort k beträgt. Daraus ergibt sich folgende Zielfunktion (R. E. Burkard 2013, 2743):

$$\min_{\varphi \in S_n} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\varphi(i)\varphi(j)} + \sum_{i=1}^{n} b_{i\varphi(i)} \quad \text{mit} \quad (2.6)$$

$$f_{ij} = \begin{cases} 1 & wenn\ Betrieb\ i\ verbunden\ ist\ mit\ Betrieb\ j \\ 0 & sonst \end{cases}$$

S_n ist dabei die Menge aller Kombinationen der möglichen Zuordnungen von Betrieben zu den Standorten. Die erste Summe beschreibt die Kosten, wenn ein Betrieb i dem Standort $\varphi(i)$ und ein Betrieb j dem Standort $\varphi(j)$ zugeordnet wird. Dabei ist der Güterfluss f_{ij} unabhängig von der Kombination aus Betrieb und Standort, da sich die Bedarfe nicht ändern. Die Distanz ist nicht unabhängig von der Standortwahl, weshalb der Permutationsparameter φ im Index benötigt wird. Die zweite Summe beschreibt die Errichtungskosten des Betriebes i am Standort $\varphi(i)$. Da wie beim WLP eine Vorauswahl zur Bestimmung der potentiellen Standorte durchgeführt wurde und sowohl der Güterfluss, als auch die Errichtungskosten bekannt und abzählbar sind, ist auch das QAP von diskreter Natur.

Alle bisher genannten Probleme gehören zur Gruppe der kombinatorischen Optimierung. Dieser Teilbereich der gemischt-ganzzahligen Optimierung beschäftigt sich mit der Suche nach einer optimalen Reihenfolge und/oder Auswahl aus einem diskretem Suchraum zur Optimierung einer Zielfunktion (Klaus, Krieger und Krupp 2012, S. 273).

Diese Auflistung von Problemstellungen in logistischen Netzwerken zeigt deutlich, wie unterschiedlich reale Herausforderungen trotz ähnlicher Strukturen sein können. Dementsprechend ist es notwendig exakte Zielfunktionen und detaillierte Nebenbedingungen zu formulieren, um die Feinheiten praktischer Probleme in mathematische Gleichungen transformieren zu können. Inwieweit diese Gleichungen durch die Anwendung der betrachteten naturbasierten Heuristiken gelöst werden können, wird in den folgenden Kapiteln untersucht.

3 Darstellung der genutzten Verfahren

Naturbasierte Algorithmen gehören zur Gruppe der populationsbasierten Verfahren. Im Gegensatz zu den zugbasierten Verfahren, welche nur eine Lösung pro Iterationsschritt betrachten, werden hierbei Lösungsmengen untersucht. Man spricht dabei auch von (Lösungs-)Populationen (Domschke und Scholl 2010, S. 31-32). Diese spezielle Gruppe von Algorithmen hat ihren Ursprung in der Natur, wobei ihre grundsätzliche Logik aus der Evolutionstheorie oder dem Verhalten von Tieren stammt.

Dieses Kapitel stellt verschiedene naturbasierte Verfahren zur Lösung mathematischer Optimierungsprobleme vor. Insbesondere wird auf ihren Ursprung in der Natur eingegangen und wie diese Logik in mathematische Modelle umgesetzt werden kann. Zudem werden ihre Entscheidungsvariablen beschrieben, sowie die Arbeitsweise der Verfahren erläutert.

3.1 Ameisenalgorithmus

3.1.1 Ursprung in der Natur

Der Ameisenalgorithmus ist ein stochastisches, metaheuristisches Verfahren der kombinatorischen Optimierung, welches auf dem Schwarmverhalten von Ameisen bei der Futtersuche basiert (Domschke und Scholl 2010, S. 36). Ein exemplarischer Verlauf der Futtersuche mit zwei möglichen Pfaden wird in **Abbildung 3-1** dargestellt. So sondern Ameisen Duftstoffe (Pheromone) als Orientierungshilfe für andere Ameisen aus, um ihnen einen Weg zur Futterquelle aufzuzeigen. Aufgrund der mangelnden Pheromonkonzentration zu Beginn der Suche wählen einzelne Ameisen zufällig einen Pfad aus, doch folgen anschließend der Spur von Pheromonen, sobald sie diese entdeckt haben und hinterlassen ihre eigene Pheromonenspur. Je mehr Ameisen dieser Spur folgen, desto intensiver wird die Pheromonenspur, was wiederum noch mehr Ameisen anlockt (Domschke und Scholl 2010, S. 36). Da sich die Duftstoffe mit der Zeit wieder verflüchtigen, ist die Konzentration der Pheromone bei gleicher Frequentierung auf dem kürzesten Weg höher, selbst wenn am Anfang die Mehrzahl an Ameisen einen längeren Weg benutzt (Gerdes, Klawonn und Kruse 2004, S. 29). Eine solche Analogie wird bei dem Ameisenalgorithmus eingesetzt.

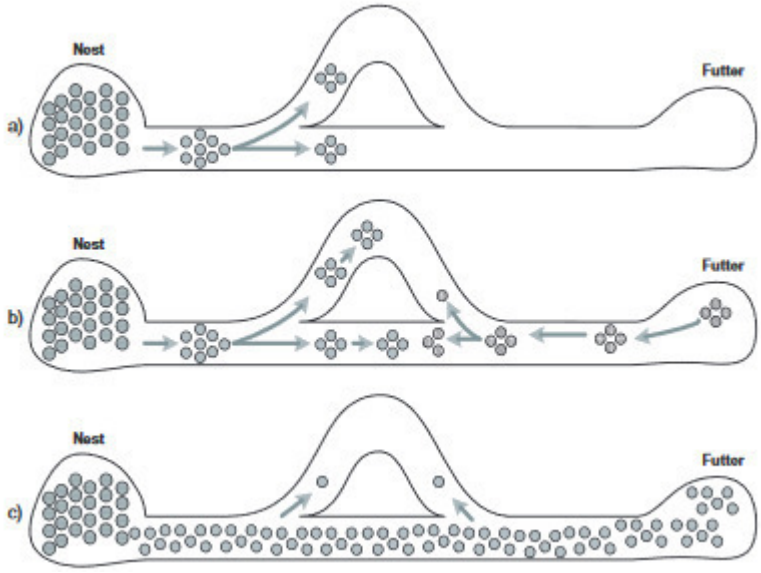


Abbildung 3-1: Schwarmintelligenz von Ameisen bei der Futtersuche (Günes, Spaniol und Kähmer 2003, S. 204)

3.1.2 Historischer Verlauf

Die Idee der Ameisenkolonieoptimierung (Ant Colony Optimization, ACO) beruht auf den Überlegungen von Dorgio, Maniezzo und Colorni (Gerdes, Klawonn und Kruse 2004, S. 29). 1991 entwickelten sie den ersten Ameisenalgorithmus unter dem Namen *Ant System* (AS) (Cordón, Herrera und Thomas 2002, S. 12) als heuristische Lösungsmöglichkeit für das Traveling Salesman Problem (Dorgio, Maniezzo und Colorni 1996, S. 29)**. Tabelle 3-1** zeigt in chronologischer Reihenfolge welche weiteren ACO-Algorithmen existieren. Im Laufe der Zeit wurden auf der Basis des AS immer weitere Ameisenalgorithmen entwickelt, oder bereits bestehende verbessert, sodass sie zu einem festen Bestandteil in der kombinatorischen Optimierung geworden sind. Hervorzuheben sind hierbei insbesondere das *Ant Colony System* (ACS) und das *MAX-MIN Ant System* (MMAS), welche zwei wichtige Erweiterungen des Ausgangsmodels darstellen und deshalb neben dem AS in den nächsten Kapiteln ausführlich erläutert werden.

Tabelle 3-1: Historische Entwicklung von Ameisenalgorithmen (Yaseen und Nada M. 2008, S. 351)

Jahr	Algorithmus	Autor
1991	Ant System	Dorgio et. al
1992	Elitist A.S.	Dorgi. et. al
1995	Ant-Q	Cambardella & Dorgio
1996	Ant Colony System	Cambardella & Dorgio
1996	Max-Min-A.S.	Stutzle & Hoos
1997	Rank Based A.S.	Bullnheimer et. al.
1999	ANTS	Maniezzo
2000	BWAS	Cordon et. al.
2001	Hyper-Cube A.S.	Blum et. al.

3.1.3 Arbeitsweise

Ameisenalgorithmen werden vor allem bei Routen- oder Tourenplanungen eingesetzt. Doch auch Maschinenbelegungen oder Personaleinsatzpläne können dadurch optimiert werden. Allgemein sind die Strukturen der Probleme, welche mithilfe von ACO-Algorithmen gelöst werden können dadurch gekennzeichnet, dass (Gerdes, Klawonn und Kruse 2004, S. 30)

- der Suchraum bzw. die möglichen Lösungen aus Sequenzen der Form (n_1, …,n_i) bestehen, wobei diese nicht die selbe Länge haben müssen
- nicht alle Sequenzen zulässige Lösungen sein müssen. Durch eine Menge Ω von Restriktionen können Sequenzen als Lösungen ausscheiden
- eine Nachbarschaftsbeziehung zwischen den Sequenzen besteht. Diese Beziehung beschreibt, wie man von einer (unvollständigen) kürzeren Sequenz zu einer längeren Sequenz gelangt, ohne dabei die Restriktionen aus Ω zu missachten
- Zudem ist es von Vorteil, wenn sich die Zielfunktion auch auf Teilsequenzen anwenden lässt

Der prinzipielle Ablauf von Ameisenalgorithmen lässt sich wie folgt beschreiben (Gerdes, Klawonn und Kruse 2004, S. 30):

- Es machen sich immer wieder neue Ameisen auf dem Weg zur Futterstelle. Ein Iterationsschritt ist dann komplett, wenn alle Ameisen nach dem Besuch der Futterstelle wieder zu ihrem Bau zurückkehren.
- An den Stellen, an denen eine Sequenz zu einer Anderen übergeht, werden Pheromone verteilt. Die Menge ist entweder von der Gesamtsequenzqualität abhängig, oder von einer Teillösung im bisherigen Schritt, sofern die Zielfunktion dies zulässt.
- Die Wahl eines bestimmten Weges ist von der Pheromonkonzentration abhängig. Je höher die Konzentration, desto wahrscheinlicher wird dieser Weg gewählt.
- Aufgrund der Verflüchtigung der Pheromone kann die Pheromonkonzentration nur dann hochgehalten werden, wenn viele Ameisen denselben Weg nehmen.

3.1.4 Mathematische Modelle

Ant System

Um die Funktionsweise des *Ant System* nachvollziehbar darstellen zu können ist eine Reihe von Vorüberlegungen und Definitionen zu treffen. Gegeben sei eine Anzahl *n* von Städten zwischen denen eine optimale Strecke im Rahmen einer Tourenplanung bestimmt werden soll. Die Distanz zwischen zwei Städten *i* und *j* beträgt d_{ij} und lässt sich durch Entfernungsmetriken berechnen. Zwei der gängigsten Metriken sind die

City-Block-Distanz $d_{ij} = |x_i - x_j| + |y_i - y_j|$ und die (3.1)

Euklidische Distanz $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ (3.2)

Bei der City-Block-Distanz werden Entfernungen überwunden, indem nur horizontale und vertikale Wege zurückgelegt werden, wohingegen bei der Euklidischen Distanz der direkte Weg, also die "Luftlinie" betrachtet wird (Melter 1987, S. 235), (Danielsson 1980, S. 227-228).

Jede Ameise aus der Gesamtmenge *m* verhält sich dabei wie folgt (Dorgio, Maniezzo und Colorni 1996, S. 31), (Focke 2006, S. 160):

- Die Wahrscheinlichkeit für die Wahl vom Knoten *i* den Knoten *j* zu besuchen ist abhängig von der Entfernung des Knotens *j* zum momentanen Standort, sowie von der Pheromonkonzentration auf der Verbindungskante (*i, j*). Formeltechnisch lässt sich diese Wahrscheinlichkeit folgenderweise darstellen (Focke 2006, S. 160-161), (Dorgio, Maniezzo und Colorni 1996, S. 31), (Naqvi und Matheru 2011, S. 162):

$$p_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{l \in J_i^k} ([\tau_{il}(t)]^{\alpha} * [\eta_{il}]^{\beta})} & wenn\ j \in J_k^i \\ 0 & sonst \end{cases} \quad (3.3)$$

  Diese Formel nennt man *Übergangsregel*. Hierbei sind α und β frei wählbare Parameter, welche den relativen Einfluss der Pheromonmenge $\tau_{ij}(t)$ auf der Kante (*i, j*) der Tour *t* und der "*heuristic desirability*" $n_{ij} = \frac{1}{d_{ij}}$ beschreibt. Der Wert η_{ij} kann als statisch angesehen werden, da sich die Distanzen zwischen den Knoten während des Durchlaufes nicht ändern. Für den Fall $\alpha = 0$ werden die Terme, welche die Pheromonmenge beinhalten zu 1 und haben somit keine Relevanz für die Berechnung der Wahrscheinlichkeit mehr. Dies hat zur Folge, dass ausschließlich die Distanzen zwischen den Knoten *i* und *j* ausschlaggebend für die Wahl eines Weges sind. Der Grundgedanke der Ameisenkolonie wäre somit außer Acht gelassen. Umgekehrt verhält es sich für den Fall $\beta = 0$. Hier verlieren die Terme mit der Information über die Distanzen ihre Bedeutung, was zur Folge hat, dass der am häufigsten genutzte Weg als Lösung des Problems ausgegeben wird, ungeachtet ob sie die optimale Tour darstellt oder nicht (Focke 2006, S. 161).

- Mittels einer Tabu-Liste J_i^k für jede Ameise *k* und jeden Knoten *i* wird sichergestellt, dass der Knoten *j* in der gleichen Tour nur einmal besucht wird
- Jede Ameise versprüht ihre Pheromone auf den genutzten Kanten (*i, j*)

Charakteristisch für das AS ist, dass die Bewertung der gefunden Wege durch eine Veränderung der Pheromonkonzentration, auch Pheromon-Update genannt, von allen

Ameisen ausgelöst wird, nachdem sie ihre Tour beendet haben (Naqvi und Matheru 2011, S. 162). Auf ihrem Weg von Knoten *i* zu Knoten *j* stößt jede Ameise gewisse Menge

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L^{k}(t)} & \text{wenn Tour t von Ameise k Kante } (i,j) \text{enthält} \\ 0 & sonst \end{cases} \quad (3.4)$$

von Pheromonen entlang der Kanten (*i, j*) des gewählten Weges aus, wobei Q ein konstanter Parameter und $\mathrm{L^k(t)}$ die Länge der von Tour t der k-ten Ameise ist (Dorgio, Maniezzo und Colorni 1996, S. 31).

Die Formel für das Pheromon-Update τ_{ij}^{k} durch den Pheromonausstoß der Ameise *k* auf den Kanten (*i, j*) kann durch

$$\tau_{ij}^{k} \leftarrow (1-\mathrm{p}) * \tau_{ij}^{k} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \quad (3.5)$$

dargestellt werden (Naqvi und Matheru 2011, S. 162). Der Term $(1-p)$ beschreibt die Verflüchtigung der bisher verteilten Pheromone, mit $p \in (0,1]$. Je höher der Wert p also gewählt wird, desto höher ist die Verdunstungsrate der Pheromone. $\sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t)$ ist die neu dazukommende Pheromonmenge aller Ameisen, welche die Kante (*i, j*) als Weg genutzt haben.

Ant Colony System

Die erste nennenswerte Erweiterung des bisherigen AS wurde im Jahr 1996 von Marco Dorgio und Luca Gambardella entwickelt und trägt den Namen *Ant Colony System* (Naqvi und Matheru 2011, S. 162).Das ACS ist der erste Ameisenalgorithmus, welcher sich eine Kandidatenliste zu Nutze machte. Jeder Knoten erhält eine Kandidatenliste mit den nächstgelegenen Knoten, welche überprüft werden sollen. Dies ist besonders bei umfangreichen Problemen hilfreich, um so den Suchraum zu verkleinern. Erst wenn alle Knoten der Liste besucht wurden, werden die übrig gebliebenen Knoten ausgewählt (Dorgio und Stützle 2004, S. 79). Die Unterschiede des ACS zum herkömmlichen AS lassen sich in drei Punkten verdeutlichen (Dorgio und Stützle 2004, S. 76) (Focke 2006, S. 165).

1) Die *Übergangsregel* liefert zwei Möglichkeiten zur Auswahl einer Kante (*i, j*). Zum einem kann das bisher erworbene Wissen genutzt werden um den optimalen Weg von Koten *i* zu Knoten *j* zu wählen. Zum anderen können neue Routen entdeckt werden, indem mit einer bestimmten Wahrscheinlichkeit in Anhängigkeit von $\tau_{\mathrm{ij}}(\mathrm{t})$ und η_{ij} neue Wege erforscht werden. Welche dieser zwei Methoden ausgewählt wird ist zufällig. Die neue *Übergangsregel* lautet somit:

$$j = \begin{cases} \arg max_{u \in j_i^k} \{[\tau_{iu}(t)] * [\eta_{iu}{}^{\beta}] \; wenn \; q \leq q_0 \\ \psi & wenn \; q > q_0 \end{cases} \quad (3.6)$$

Wobei q eine gleichverteile Zufallszahl zwischen [0, 1] und q_0 mit $(0 \leq q_0 \leq 1)$ ein wählbarer Parameter ist. Dadurch wird erreicht, dass, wie eben erwähnt, zufällig eine der beiden Methoden zur Bestimmung des nächsten Knotens angewandt wird. Ψ ist ein zufälliger Knoten dessen Auswahl mit der Wahrscheinlichkeit

$$p_{ij}^{k}(t)=\frac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{l \in J_i^k}([\tau_{il}(t)]^{\alpha} * [\eta_{il}]^{\beta})} \quad (3.7)$$

ähnlich wie in (3.3) berechnet wird. Die Wahrscheinlichkeit, dass der nächstbeste Knoten vom aktuellem Standort aus gewählt wird, beträgt also q_0. Mit einer Wahrscheinlichkeit von $1 - q_0$ werden neue Kanten erforscht. Durch die Variation des Parameters q_0 kann dementsprechend geregelt werden, ob der Algorithmus die bisher bestmögliche Lösung ermitteln soll, oder ob dieser neue Touren finden soll, auf deren Basis eventuell eine noch bessere Lösung erzielt werden kann. (Dorgio und Stützle 2004, S. 77).

2) Im Gegensatz zum AS dürfen beim ACS nicht alle Ameisen eine Pheromonspur legen, sondern nur die Ameise, welche die bislang beste Route gefunden hat (Dorgio und Stützle 2004, S. 77). Aufgrund der weniger stark verteilten Pheromonspuren wählen die Ameisen deshalb vornehmlich Routen in der Nähe der bislang besten Tour (Focke 2006, S. 166). Die *globale Pheromon-Update-Regel* für diesen Fall lässt sich folgendermaßen darstellen:

$$\tau_{ij}(t) \leftarrow (1-p) * \tau_{ij}(t) + p * \Delta\tau_{ij}(t) \quad (3.8)$$

p steht hierbei wieder für die Verflüchtigungsrate der Pheromone pro Iteration und es gilt:

$$\Delta\tau_{ij}(t) = \frac{1}{L^*} \text{ mit } L^* \text{ als Länge der besten Tour } T^* \quad (3.9)$$

Angemerkt sei, dass die Kante (i, j) ebenfalls zur bisher besten Tour T^* gehören muss (Focke 2006, S. 166).

3) Um die Erforschung neuer Wege zu fördern, wird jedes Mal, wenn die Kante (*i, j*) von einer Ameise als Weg benutzt wird, eine gewisse Menge von Pheromonen auf dieser Kante entfernt (Dorgio und Stützle 2004, S. 76-77). Dies ist notwendig, um nicht Gefahr zu laufen, dass die Ameisen sich zu sehr auf die Lösungen in der Nähe der optimalen Route fokussieren und so unter Umständen ein lokales Optimum nicht mehr verlassen können (Focke 2006, S. 166). Der Wert, der reduzierten Pheromonenkonzentration ist durch das *lokale Pheromon-Update*

$$\tau_{ij}(t) \leftarrow (1-\varepsilon) * \tau_{ij}(t) + \varepsilon * \tau_0 \quad (3.10)$$

bestimmt. ε ist hierbei wieder ein wählbarer Parameter zwischen 0 und 1, wobei sich ein Wert von $\varepsilon = 0{,}1$ versuchsweise als guter Wert für Berechnungen herausgestellt hat. Zudem erwies sich $\tau_0 = \frac{1}{n * C^{nm}}$, mit der Anzahl der zur besuchenden Städte *n* und C^{nm} als die Länge der nearest-neighbour-Tour als guter Wert für die anfängliche Pheromonenkonzentration (Dorgio und Stützle 2004, S. 78).

MAX-MIN-Ant System

Eine weitere Verbesserung des AS wurde 1996 von Thomas Stützle und Holger H. Hoos entwickelt. Beim MAX-MIN-Ant System (MMAS) ist es nur der Ameise erlaubt die Pheromonkonzentration zu aktualisieren, welche die bis dahin kürzeste Route gefunden hat. Ähnlich wie beim ACS besteht dadurch jedoch das Risiko sich zu sehr auf die Wege in der Nähe der momentan besten Route zu fokussieren und somit in einem lokalem Optimum zu stagnieren. Um dieser Gefahr entgegenzuwirken wird die Pheromonkonzentration durch eine untere Schranke τ_{min}und einer oberen Schranke τ_{max} begrenzt (Dorgio und Stützle 2004, S. 74). Damit wird erreicht, dass selten genutzte Wege zumindest mit einer geringen Wahrscheinlichkeit gewählt werden, um so die Erforschung neuer Wege zu fördern (Focke 2006, S. 173). τ_{max} wird durch den Kehrwert des Produktes aus der Wahrscheinlichkeit p und der Länge der bisher besten Route d^{bs} bestimmt. Wird im Laufe der Lösungssuche eine neue beste Route gefunden, so aktualisiert sich auch τ_{max} (Dorgio und Stützle 2004, S. 75). Desweiteren beginnen alle Kanten mit der maximalen Pheromonenmenge τ_{max}, welche in jeder Iteration um die Verdunstungsrate p verringert wird (Dorgio und Stützle 2004, S. 74). Sollte dennoch nach einer festgelegten Anzahl von Iterationen keine verbesserte Tour gefunden werden und somit eine Stagnation drohen, werden die Pheromonkonzentrationen auf den Kanten durch den "trail-smoothing-Effekt" proportional zum Wert $[\tau_{max} - \tau_{ij}(t)]$ erhöht, wobei $\tau_{ij}(t)$ die momentane Pheromonkonzentration ist (Shtovba 2005, S. 173).

Die *Pheromon-Update-Regel* für das MMAS lautet somit:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau^{best} \text{ , mit } \Delta\tau^{best} = \frac{1}{d^{best}} \qquad (3.11)$$

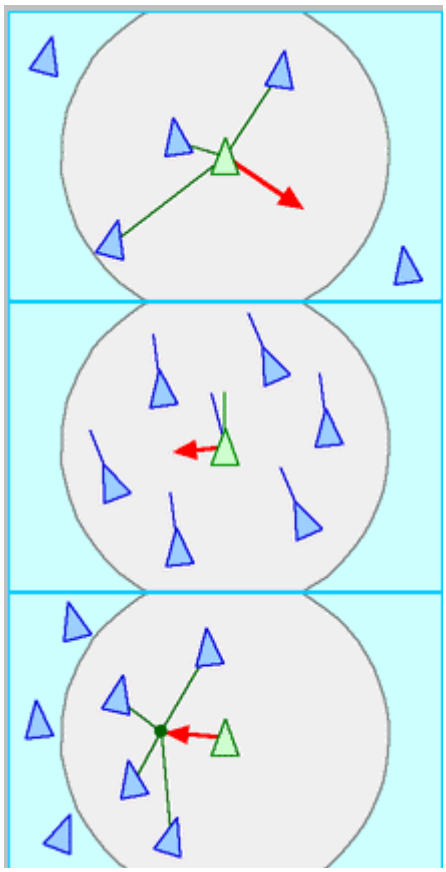
Die Pheromonkonzentration wird entweder von der Ameise mit der bisher besten Route, oder mit der besten Route der momentanen Iteration aktualisiert. In der Regel werden die Methoden abwechselnd eingesetzt, wobei man bestimmen kann welche Methode bevorzugt angewendet werden soll. Versuchsreihen haben ergeben, dass für kleine TSP der Fokus auf den iterationsbesten Routen liegen soll, wobei sich der Einsatz der bisher besten Tour bei großen TSP mit mehreren hundert Knoten bewährt hat (Dorgio und Stützle 2004, S. 75).

3.2 Partikelschwarmoptimierung

3.2.1 Ursprung in der Natur

Der Ursprung der Partikelschwarmoptimierung (PSO) liegt in dem Verhalten von Tierschwärmen. So bewegen sich beispielsweise Vögel oder Wanderheuschrecken in

Schwärmen stets nach einem ähnlichem Verhaltensmuster. Angetrieben von der Gefahr vom Hintermann gefressen zu werden, müssen Wanderheuschrecken bei der Futtersuche stets in Bewegung bleiben, um so einen Abstand zu ihren Artgenossen aufrecht erhalten zu können. Durch eine "serotoningesteuerte Geselligkeit", welche durch positive Rückkopplung überproportional zur Anzahl der Artgenossen steigt, bleibt der Schwarm zusammen und es wird verhindert, dass sich die Individuen in alle Richtungen verstreuen (Fisher 2010, S. 39). Der erste, der dieses Schwarmverhalten in einem Modell umsetze war Craig Reynolds. Er entwickelte 1986 sogenannte Boids, welche die Tiere innerhalb eines Schwarms simulieren (Fisher 2010, S. 39-40). Laut Reynolds folgen die Boids drei einfachen Gesetzmäßigkeiten (siehe **Abbildung 3-2**) (Fisher 2010, S. 40), (Reynolds 2001):



Abstoßung: Vermeidung von Zusammenstöße mit anderen Artgenossen

Ausrichtung: Bewegung in die durchschnittliche Richtung der Nachbarn

Anziehung: Bewegung zur durchschnittlichen Position der Nachbarn

Abbildung 3-2: Verhalten der Boids im Boid-Modell (Reynolds 2001)

3.2.2 Arbeitsweise

Das Modell von Reynolds machten sich James Kennedy und Russel Eberhart zu Nutze und entwickelten auf dessen Basis das Grundmodell der Partikelschwarmoptimierung (Clerc 2006, S. 87). Dieses Verfahren wurde im Jahr 1995 als Lösungsmethode für die Optimierung kontinuierlicher nichtlinearer Funktionen vorgestellt (Eberhart und Kennedy 1995, S. 39).

Bei der PSO werden die potentiellen Lösungskandidaten *i*, im weiteren Verlauf *Partikel* genannt, durch die einzelnen Individuen eines Schwarms symbolisiert und bewegen sich frei durch einen definierten Suchraum *S*. In Anlehnung an das Modell von Reynolds basiert das

Verhalten der Partikel nach Kennedy und Eberhart auf folgenden Prinzipen (Kennedy, Eberhart und Shi 2001, S. 288-289):

- **Evaluation**: Selbst die kleinsten lebenden Organismen sind in der Lage Umwelteinflüsse bewerten zu können
- **Vergleich**: Individuen eines Schwarm vergleichen sich mit ihren Nachbarn was den Drang nach Verbesserung fördert
- **Imitation**: Individuen ahmen erfolgreichere Artgenossen nach

Für die Partikel der PSO bedeutet dies, dass sie ihre aktuelle Belegung mittels eines Fitnesswertes angeben können (Evaluation), ihre Fitnesswerte mit anderen Partikeln vergleichen, wodurch die Güte des Wertes bestimmt wird (Vergleich) und, dass sie den besten Fitnesswert zu einem gewissen Teil in ihre Berechnungen zur nächsten Belegung einfließen lassen (Imitation) (Bogon 2012, S. 39). Jedes Partikel *s* ist mit einer Position *p* innerhalb des Suchraumes belegt und bewegt sich mit einer Geschwindigkeit *v*. Diese beiden Werte können mittels einer Funktion *f,* der Fitnessfunktion, abgebildet werden (Bogon 2012, S. 40). Desweiteren besitzt jedes Partikel ein Gedächtnis und kennt seine bisher beste Position (Fitness), welche mit p_{best} deklariert wird. Auch die global beste Lösung aller Partikel g_{best}ist den Partikeln bekannt (Eberhart und Kennedy 1995, S. 39). Anstelle der global besten Position kann auch mit der lokalen besten Position l_{best} gearbeitet werden. Dabei ordnet man den Partikeln ihre *k* nächsten Nachbarn zu, zwischen denen das lokale Optimum bestimmt wird. Bei $k = 2$ Nachbarn für Partikel *i* besteht die Nachbarschaft aus den Partikeln *i - 1* und *i + 1* und führt zu einer ringförmigen Struktur wie in **Abbildung 3-3** zu sehen (Kennedy, Eberhart und Shi 2001, S. 290).

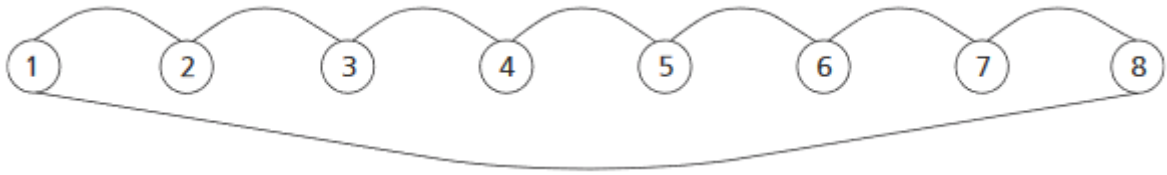


Abbildung 3-3:Ringstruktur der l_{best} -Nachbarschaft mit k = 2 (Kennedy, Eberhart und Shi 2001, S. 291)

Je größer der Wert *k* für die l_{best}-Nachbarschaft gewählt wird, umso stärker nimmt es die Struktur der g_{best}-Nachbarschaft aus **Abbildung 3-4** an (Bogon 2012, S. 40).

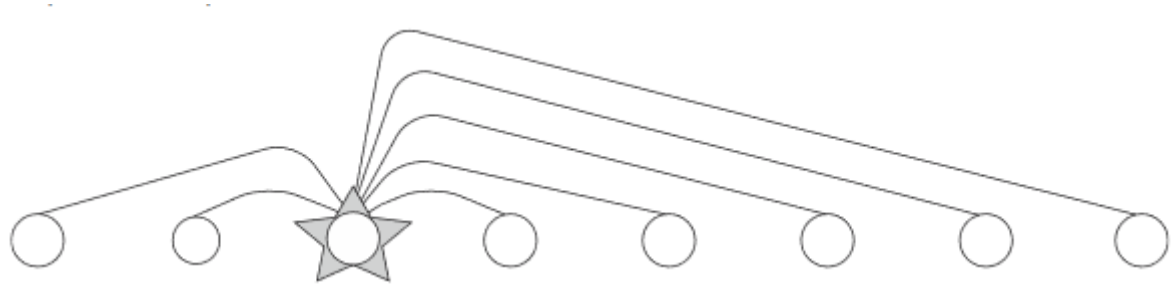


Abbildung 3-4: Struktur der g_{best}-Nachbarschaft wenn i=3 den besten Fitnesswert besitzt (Kennedy, Eberhart und Shi 2001, S. 291)

Die Auswahl der Methode hat wesentlichen Einfluss auf das Verhalten der Partikel. So wird bei der g_{best}-Nachbarschaft Methode die global beste Position schnell verbreitet, was zu einem konvergenten und weniger explorativen Verhalten des Schwarms führt. Dadurch werden die Partikel jedoch von lokalen Optima angezogen, weshalb es länger dauert, bis ein besseres Ergebnis erzielt wird (Bogon 2012, S. 42), (Kennedy, Eberhart und Shi 2001, S. 290-292). Bei der l_{best} -Nachbarschaft Methode fokussiert der Schwarm nicht so schnell, weswegen mehr Position im Suchraum abgesucht werden. Dies hat jedoch den Nachteil, dass es dementsprechend Länger dauert, bis eine Fokussierung auf ein Optimum vollzogen wird (Kennedy und Mendes 2002, S. 1673-1674), (Bogon 2012, S. 41).

3.2.3 Mathematische Modelle

Der Ablauf der PSO lässt sich nun wie folgt beschreiben (Bogon 2012, S. 40-41), (Eberhart und Shi 2007, S. 88), (Kennedy, Eberhart und Shi 2001, S. 295):

Die Partikel starten mit einer Ausgangsposition $p_0$und einer Ausgangsgeschwindigkeit v_0 in einen definiertem Suchraum *S.* Die Postionen und Geschwindigkeiten der einzelnen Partikel heißen Fitnesswerte und werden in jedem Iterationsschritt neu berechnet. Jedes Partikel vergleicht daraufhin seinen aktuellen Fitnesswert mit seinem bisher bestem Wert p_{best} und ersetz diesen, sofern sein aktueller Wert besser ist. Danach wird der Fitnesswert mit der lokalen bzw. globalen (je nach Wahl der Methode) besten Position verglichen, welche ebenfalls ersetz werden, falls der momentane Wert besser ist. Abschließend wird die neue Position des Partikels innerhalb des Suchraumes mit

$$v_t = v_{t-1} + r_1 * c_1(p_{best} - p_{actual}) + r_2 * c_2(g_{best} - p_{actual}) \text{ und} \quad (3.12)$$

$$p_{new} = p_{old} + v_t \quad (3.13)$$

berechnet. $c_1$und $c_2$sind dabei konstante Parameter, welche den Einfluss des persönlichen und globalen Wissens steuern und r ein Zufallswert zwischen 0 und 1 (Bogon 2012, S. 41). Diese aktualisierten Fitnesswerte werden wiederum mit p_{best} und g_{best}bzw. l_{best} verglichen und somit eine neue Iteration gestartet bis ein ausreichen guter Fitnesswert gefunden, oder die maximale Anzahl von Iteration erreicht wurde. Da die Änderung der Geschwindigkeit stochastischen Merkmalen unterliegt, besteht die Gefahr, dass sie immer weiter wächst und so die Bewegungsbahn der Partikel ebenfalls immer größer wird bis sie unter Umständen

gegen unendlich tangiert. Um dies zu verhindert wird eine maximale Geschwindigkeit V_{max} festgelegt (Kennedy, Eberhart und Shi 2001, S. 328-329). Dabei darf V_{max} auch nicht zu klein gewählt werden, da ansonsten die Partikel einen zu kleinen Raum absuchen und somit in einem lokalem Optimum stagnieren (Eberhart und Shi 2007, S. 89).

Im allgemeinen wird zwischen zwei Möglichkeiten zur Berechnung der Bewegungen unterschieden. Die gängigste Methode ist die "inertia weight"-Methode. Hierbei wird der alten Geschwindigkeit eine Trägheit *w* zugeordnet, dessen Gewichtung die Konvergenz des Algorithmus steuert (Bogon 2012, S. 41), (Eberhart und Shi 2007, S. 89-90). Somit gilt für die Berechnung der Geschwindigkeit folgende Formel:

$$v_t = w * v_{t-1} + r_1 * c_1(p_{best} - p_{actual}) + r_2 * c_2(g_{best} - p_{actual}) \quad (3.14)$$

Der einzige Unterschied zu (3.12) ist der angesprochene Gewichtungsfaktor *w* der vorherigen Geschwindigkeit. Die neue Position kann analog zu (3.13) berechnet werden, da dort keine Änderungen vorgenommen wurden.

Die zweite Möglichkeit zur Bewegungsberechnung der Partikel ist die "contruction"-Formel. Dabei werden Parameter der "inertia weight"-Formel zusammengefasst und mittels einem Parameter abgebildet (Bogon 2012, S. 41). Um die Konvergenz der PSO zu fördern wurde eine neue Formel zur Berechnung der Geschwindigkeit entwickelt:

$$X = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\vartheta}|} \text{ mit } \varphi = c_1 + c_2 \text{ und } \varphi > 4 \quad (3.15)$$

$$v_t = X\,[v_{t-1} + r_1 * c_1(p_{best} - p_{actual}) + r_2 * c_2(g_{best} - p_{actual})] \quad (3.16)$$

Im nächsten Iterationsschritt bilden die aktualisierten Geschwindigkeiten die Vergleichsgrundlage zur Berechnung weiterer Geschwindigkeiten.

Die Formeln (3.12), (3.14) und (3.16) gelten für die g_{best}-Nachbarschaft-Methode. Soll die Geschwindigkeit auf Basis der lokalen besten Position berechnet werden, so wird g_{best} durch l_{best} ersetzt.

3.3 Evolutionärer Algorithmus

3.3.1 Ursprung in der Natur

Im Gegensatz zu den beiden bisher genannten Verfahren basiert der Evolutionäre Algorithmus (EA) nicht auf dem Verhalten von Tierschwärmen, sondern ist an die Prinzipen der biologischen Evolution nach Charles Darwin und Alfred Wallace angelehnt. Grundlage ist die Annahme, dass Lebewesen in der Lage sind sich durch Innovationen an die verändernde Umwelt anzupassen und dies an ihre Nachkommen weitergeben können (Gerdes, Klawonn und Kruse 2004, S. 217). Um die nachfolgenden Erläuterungen zu evolutionären Algorithmen besser nachvollziehen zu können, ist eine grundlegende Kenntnis der Terminologie der biologischen Evolution notwendig. In **Tabelle 3-2** werden deshalb die wichtigsten Begriffe erläutert und der Bezug zu evolutionären Algorithmen dargestellt.

Tabelle 3-2: Begriffe der Genetik im Zusammenhang mit evolutionären Algorithmen (Gerdes, Klawonn und Kruse 2004, S. 34)

Name	Biologische Evolution	Evolutionäre Algorithmen
Chromosom	legt den Bauplan bzw. die Eigenschaften eines Lebewesens (bzw. eines Teils davon) fest	String, Zeichen- oder Zahlenkette nicht notwendig festgelegter Länge
Gen	einzelnes Teilstück eines Chromosoms, das eine "Teileigenschaft" festlegt, z.B. "Ein Gen für die Augenfarbe blau"	Character, Feature, Zeichen, Variable
Allel	Ausprägung eines Gens (z.B. Augenfarbe blau, grün, …)	Wert eines Zeichens
Locus	Ort eines Gens im Chromosom	Position eines Zeichens
Phänotyp	Erscheinungsbild des Lebewesens	Chromosom-kodierte Problemlösung
Genotyp	Das Erscheinungsbild des Chromosoms	Die Formale Kodierung einer Lösung
Population	Menge von Lebewesen	Menge bzw. Familie (Bag) von Chromosomen
Fitness	Überlebens- / Vermehrungsfähigkeit eines Lebewesens	Güte einer durch ein Chromosom kodierten Lösung
Generation	Population zu einem Zeitpunkt	Analog
Reproduktion	Erzeugen von Nach-kommen aus einem oder mehreren Lebewesen	Erzeugen von Chromosomen aus einem oder mehreren Chromosomen

3.3.2 Arbeitsweise

Die erste erfolgreiche Anwendung evolutionärer Algorithmen geht auf den Statistiker George Box zurück. Er entwickelte in den 1950er Jahren ein Verfahren namens "evolutionary operation" (EVOP) zur Optimierung des laufenden Prozesses in Chemiefabriken (Fogel 2005, S. 59). Dieses Verfahren bildete die Ausgangslage für die Entwicklung evolutionärer Algorithmen. Dabei etablierten sich insbesondere vier Typen von Algorithmen, welche bis heute eingesetzt werden:

- Genetische Algorithmen (Holland, 1962)
- Evolutionsstrategien (Rechenberg, Schwefel, 1969)
- Evolutionäre Programmierung (Fogel, Owens, Walsh, 1965)

- Genetische Programmierung (Koza, 1994)

Bevor auf diese Algorithmen im Detail eingegangen wird, ist zunächst der allgemeine Ablauf evolutionärer Algorithmen zu beschreiben. Die Ablaufstruktur eines EA ist in **Abbildung 3-5** dargestellt.

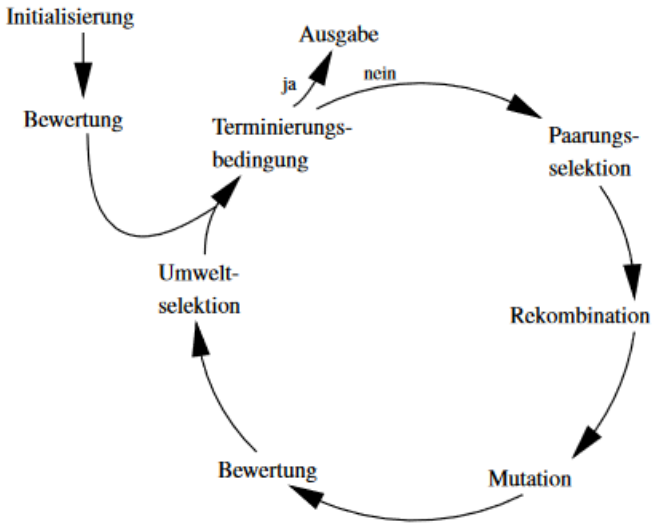


Abbildung 3-5: Struktur eines evolutionären Algorithmus (Weicker 2015, S. 25)

Jeder EA startet mit der Initialisierungsphase. Dort wird eine zufällige Anfangspopulation von Lösungskandidaten, auch Individuen genannt, festgelegt und die Güte der Individuen mittels einer Fitnessfunktion f bestimmt (Weicker 2015, S. 24). Anhand dieser Güte wird während der Paarungselektion festgelegt wie viele Nachkommen jedes Individuum erzeugt und welche Individuen für die Fortpflanzung ausscheiden (Weicker 2015, S. 24), (Pohlheim 2000, S. 13). Durch Eröffnungsheuristiken können bereits gute Anfangsindividuen erzeugt werden, wobei dadurch die Gefahr der vorzeitigen Konvergenz besteht (Gerdes, Klawonn und Kruse 2004, S. 37). Den Vorgang der Nachkommenserzeugung nennt man Rekombination. Hierbei werden die Informationen der Eltern kombiniert und auf dieser Basis neue Individuen erschaffen. Dabei kann es jedoch mit einer geringen Wahrscheinlichkeit vorkommen, dass zufällige Veränderungen beim Kind vorkommen, sodass dieses Informationen besitzt, die keinem der Elternteile zugeordnet werden können. In diesem Fall spricht man von Mutation (Pohlheim 2000, S. 15). Die neuen Individuen werden ebenfalls bewertet und in die Population der Eltern integriert (Umweltselektion). Dabei kann es vorkommen, dass aufgrund der eingeschränkten Populationsgröße, die alten Individuen durch die Neuen ersetzt werden (Weicker 2015, S. 25). Die neue Population nach jeder Iteration nennt man Generation. Am Ende jeder Iteration wird durch eine Abbruchbedingung überprüft, ob das Ziel erreicht wurde, oder ob weitere Wiederholungen notwendig sind. Dies kann beispielweise eine maximale Anzahl von Iterationen, das Erreichen einer bestimmten Güte oder eine Anzahl von Generationen ohne Verbesserung sein (Weicker 2015, S. 25).

3.3.3 Mathematische Modelle

Genetische Algorithmen

Der genetische Algorithmus ist ein stochastisches Verfahren, welche den Suchraum an mehreren Stellen gleichzeitig absucht (Domschke und Scholl 2010, S. 32). Der Ablauf eines genetischen Algorithmus (GA) ist in Anlehnung an die allgemeine Struktur evolutionärer Algorithmen in **Abbildung 3-6** dargestellt.

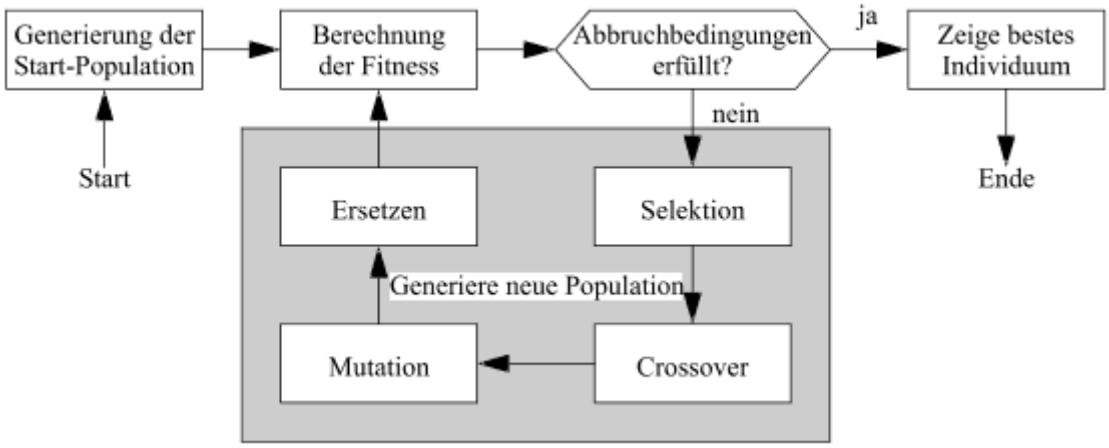


Abbildung 3-6: Ablauf eines genetischen Algorithmus (Buttelmann und Lohmann 2004, S. 153)

Bei dem GA sind die Erbinformationen eines Individuums innerhalb des Suchraumes *S* auf einem Chromosom *a* gespeichert (Gerdes, Klawonn und Kruse 2004, S. 36). Diese Chromosomen sind in Gene aufgeilt, welche binär codierte Informationen enthalten. Dabei müssen nicht zwangsläufig alle Gene die selbe Länge besitzen, wie **Abbildung 3-7** zeigt (Buttelmann und Lohmann 2004, S. 153).

$$\mathbf{a}_i = [\cdots 10|\underbrace{11001101}_{\text{Objekt-parameter } x_j}|\underbrace{011010101101}_{\text{Objekt-parameter } x_{j+1}}|01\cdots]$$

Abbildung 3-7: Binär codierter Informationen unterschiedlicher Länge eines Chromosoms a_i (Buttelmann und Lohmann 2004, S. 153)

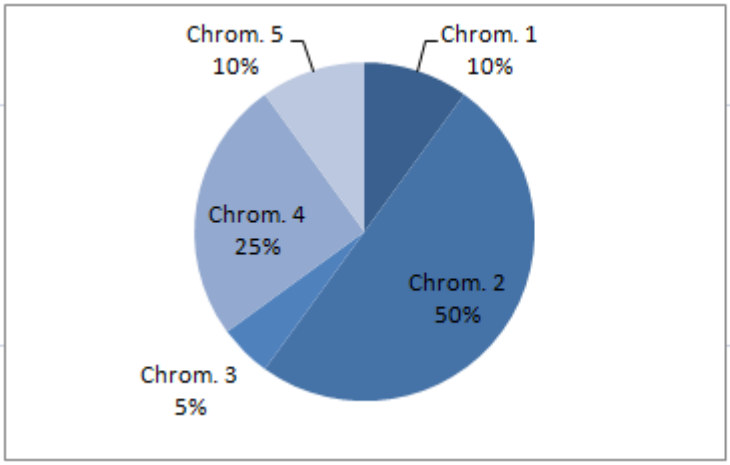
Ausgehend von den Fitnesswerten *fit* der Individuen aus der Startpopulation s werden nach dem Prinzip "survival of the fittest" die Individuen zur Nachkommenserzeugung ausgewählt. Das bedeutet, dass Individuen mit hohen Fitnesswerten eine größere Wahrscheinlichkeit haben sich fortzupflanzen, als welche mit niedrigerem Wert.

Die Wahrscheinlichkeit $p(a_0)$ zur Fortpflanzung ausgewählt zu werden wird folgendermaßen bestimmt:

$$p(a_0) = \frac{fit(a_0)}{\sum_{a \in P(t)} fit(a)} \tag{3.17}$$

Der Zähler beschreibt hierbei den Fitnesswert eines einzelnen Chromosoms a_0, welcher durch Summe der Fitnesswerte der gesamten Population $P(t)$ zum Zeitpunkt t dividiert wird. Sollte die Gesamtfitness und somit der Nenner 0 betragen, erhalten alle Chromosomen die gleiche Auswahlwahrscheinlichkeit $p(c) = \frac{1}{s}$, wobei s die Anzahl der Chromosomen pro Generation ist (Gerdes, Klawonn und Kruse 2004, S. 38).

Mit dieser Wahrscheinlichkeit arbeitet die "Roulette-Selektion", welche die am häufigsten genutzte Selektionsmethode ist. Dabei wird jedem Chromosom a, proportional zu seinem Fitnesswert, ein Segment auf einem Roulette-Rad zugeordnet(siehe **Abbildung 3-8**). Somit ist die Wahrscheinlichkeit bei Drehung des Rades ausgewählt zu werden höher, je größer der Fitnesswert und somit der zugeordnete Anteil auf dem Rad ist (Buttelmann und Lohmann 2004, S. 154).



Chromosom	eval (a)	fit (a)
1	30	0.1
2	15	0.5
3	15	0.05
4	75	0.25
5	30	0.1

Abbildung 3-8: Verteilung von Chromosomen auf einem Roulette-Rad in Abhängigkeit ihrer Fitnesswerte, in Anlehnung an (Gerdes, Klawonn und Kruse 2004, S. 39)

Es wird s-mal ein Chromosom der Vorgängerpopulation $P(t\ -1)$ mit der Wahrscheinlichkeit $p(a)$ ausgewählt und anschließend in der Hilfsmenge $P^*(t)$ gespeichert (Gerdes, Klawonn und Kruse 2004, S. 39). [6]

Um nun aus den ausgewählten Chromosomen Nachkommen zu bilden, werden die genetischen Operatoren *Rekombination* und *Mutation* benötigt. Die Rekombination binärer Variablen stellt einen Sonderfall dar und wird Crossover genannt (Pohlheim 2000, S. 39). Beim Crossover entstehen zwei neue Chromosomen durch den Austausch von Teilketten zweier Chromosomen untereinander. Welche Chromosomen der Menge $P^*(t)$ miteinander gekreuzt werden hängt von der Wahrscheinlichkeit p_a ab. Es existieren mehrere Möglichkeiten wie das Crossover vonstattengeht. Beim *Single-Point-Crossover* wird nur eine Schnittposition für den Austausch festgelegt, wohingegen es beim *Multi-Point-Crossover* mehrere Schnittpunkte geben kann, welche der Größe nach sortiert werden (siehe **Abbildung 3-9**) (Pohlheim 2000, S. 40). Werden keine Schnittpunkte verwendet, sondern

[6] Weitere Selektionsverfahren sind zu finden in (Gerdes, Klawonn und Kruse 2004, S. 79-86) und (Pohlheim 2000, S. 24-31)

wird jedes einzelne Allel untersucht und mit der Wahrscheinlichkeit $p_c = 0{,}5$ ausgetauscht, wird von *Uniform Crossover* gesprochen (Gerdes, Klawonn und Kruse 2004, S. 89).

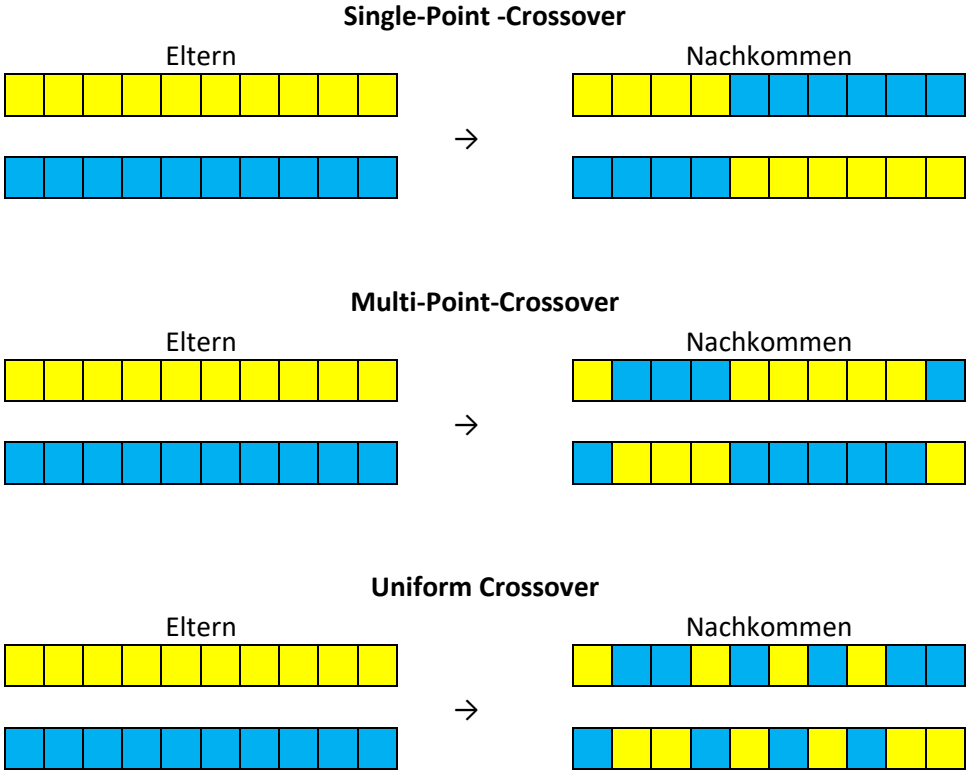


Abbildung 3-9: Single-Point, Multi-Point und Uniform Crossover im Vergleich, in Anlehnung an (Pohlheim 2000, S. 40) und (Gerdes, Klawonn und Kruse 2004, S. 89)

Bei der Mutation eines Gens wird mit einer geringen Wahrscheinlichkeit gleichverteilt zufällig ein Allel verändert (siehe **Abbildung 3-10**). Da bei der binären Codierung nur zwei Allele (0 und 1) existieren, kommt die Mutation einem Austausch gleich (Gerdes, Klawonn und Kruse 2004, S. 41).

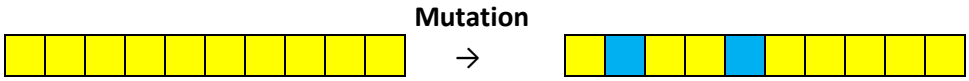


Abbildung 3-10: Mutation eines Chromosoms, in Anlehnung an (Gerdes, Klawonn und Kruse 2004, S. 41)

Anschließend werden die Individuen aus der Menge $\mathrm{P}^*(\mathrm{t})$ zur neuen Population $\mathrm{P}(\mathrm{t})$ und bildet somit in der nächsten Iteration die Basis zur Bestimmung der Population $\mathrm{P}(\mathrm{t}+1)$.

Evolutionsstrategien

Evolutionsstrategien (ES) wurden in den 1960er von Rechenberg und Schwefel zur Optimierung technisch-physikalischer Probleme mit reellwertiger Zielfunktion $f : \mathbb{R}^n \to \mathbb{R}$, mit n zu optimierenden Parametern entwickelt (Gerdes, Klawonn und Kruse 2004, S. 115).

Wie beim GA starten auch ES mit einer bewerteten Startpopulation P(0) der Ausgangspopulationsgröße μ. Das besondere bei ES ist, dass zur Bestimmung der Nachfolgepopulation $P(t+1)$ aus der Population $P(t)$, Mutationen der entscheidende Operator sind, auf welche im späteren Verlauf noch genauer eingegangen wird (Weicker 2015, S. 134). Die Anzahl von Nachkommen der Chromosomen wird mit λ deklariert, wobei bei der Auswahl der Eltern kein spezielles Selektionsverfahren angewendet wird. Ungeachtet ihrer Fitnesswerte haben alle Chromosomen die selbe Wahrscheinlichkeit ausgewählt zu werden. Stattdessen werden während der Umweltselektion nach dem *Eliteprinzip* nur die μ besten Chromosomen in die Nachfolgepopulation übernommen (Gerdes, Klawonn und Kruse 2004, S. 116). Besteht die Nachfolgepopulation nur aus den μ besten Nachkommen, spricht man von der Komma-Strategie, oder auch (μ, λ)-ES. Desweiteren existiert noch die Plus-Strategie, (μ + λ)-ES, bei der die Nachfolgepopulation aus den Eltern sowie deren Kindern besteht (siehe **Abbildung 3-11**). Teilweise ist auch ein alternierender Einsatz der beiden Verfahren möglich (Gerdes, Klawonn und Kruse 2004, S. 116-117).

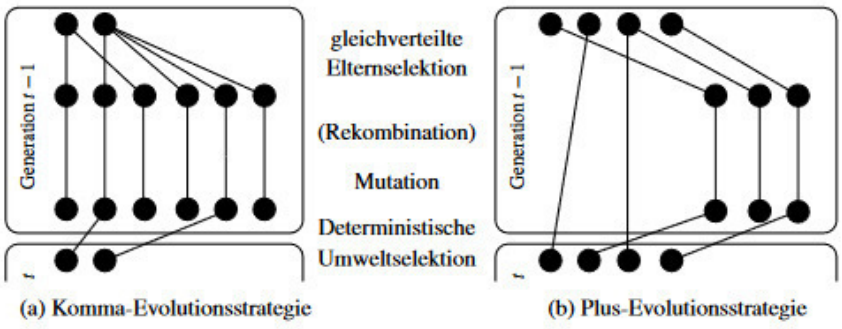


Abbildung 3-11: Bildung einer Nachfolgepopulation bei (μ, λ)-ES und (μ + λ)-ES (Weicker 2015, S. 137)

Bei der bereits erwähnten Mutation wird zu dem Allel jedes Gens nach einer Normalverteilung mit dem Erwartungswert 0 und einer Streuung σ eine Zufallszahl addiert (Gerdes, Klawonn und Kruse 2004, S. 115). Die Streuung σ ist dabei der Schrittweitenparameter der Mutation und sollte so klein gewählt werden, dass die Kinder sich nicht zu sehr von ihren Eltern unterscheiden, da ansonsten die neuen Chromosomen zufällig verstreut im Suchraum liegen (Weicker 2015, S. 134), (Gerdes, Klawonn und Kruse 2004, S. 116). Die Wahl einer geeigneten Schrittweite gestaltet sich jedoch schwierig, weshalb Anpassungen an σ vorgenommen werden müssen. Eine solche Anpassungsmöglichkeit liefert *Reichenbergs 1/5-Regel*. Sie ermittelt durch statistische Erhebungen in den letzen Generationen einen neuen Wert für σ und besagt, dass ein guter Wert gefunden ist, wenn etwa ein Fünftel der Mutationen erfolgreich ist (Weicker 2015, S. 134), (Gerdes, Klawonn und Kruse 2004, S. 118).

Durch gestiegene Rechenleistung finden immer häufiger populationsbasierte $(\mu \overset{+}{,} \lambda)$-ES ihre Anwendungen, wobei auch die Rekombination eingesetzt wird (Weicker 2015, S. 137). Häufig wird das uninformierte Crossover verwendet, welches im Rahmen der ES diskrete *Rekombination genannt* wird. Aufgrund der reellwertigen Codierung der Gene ist auch die intermediäre Rekombination eine Möglichkeit für den Informationsaustausch zweiter

Chromosomen. Dabei werden die Mittelwerte der Allele der Eltern an die Nachkommen weitergegeben (Gerdes, Klawonn und Kruse 2004, S. 122).

Evolutionäre Programmierung

Die evolutionäre Programmierung (EP) wurde ursprünglich für diskrete Zeitreihenprognosen endlicher Automaten entwickelt und stellt einen Sonderfall evolutionärer Algorithmen dar. Bei dieser Methode wird die Evolution von einem phänotypischen Gesichtspunkt aus betrachtet, was bedeutet, dass kein Rekombinationsparameter eingesetzt wird und die Änderungen der Chromosome allein durch Mutationen ausgelöst werden (Weicker 2015, S. 140), (Gerdes, Klawonn und Kruse 2004, S. 125).

Der in **Abbildung 3-12** dargestellte Ablauf evolutionärer Programme ist analog zu den Evolutionsstrategien. Auch hier sind die Chromosomen reelle Vektoren dessen Werte für die Startpopulation zufällig bestimmt werden. Dabei wird während der Initialisierungsphase jedoch sichergestellt, dass sich die Werte mittels einer Gleichverteilung in einem vorgegebenem Intervall bewegen (Gerdes, Klawonn und Kruse 2004, S. 125).

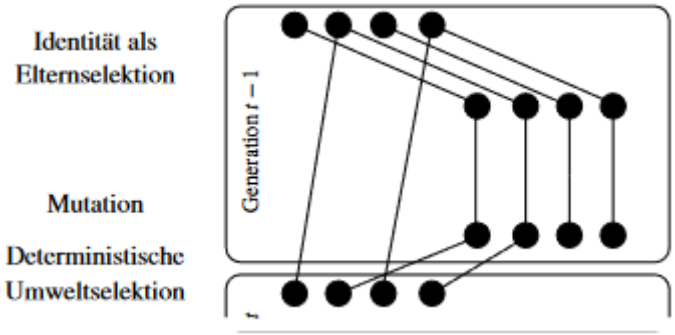


Abbildung 3-12: Ablauf der evolutionären Programmierung (Weicker 2015, S. 145)

Zunächst wird von jedem Chromosom a der Population P(t) eine Kopie erstellt, woraus dementsprechend eine neue, doppelt so große Population P*(t) gebildet wird. Danach wird auf die neue Population P*(t) ein Mutationsparameter angewendet, welcher sich in Abhängigkeit von der Fitness des Elternchromosoms und einem normalverteiltem Zufallswert:

$$\forall\, i \in \{1, \dots, s\} : c_{ij}^{*} = c_{ij} + \sqrt{k_i * fit(a_i) + z_j} * N(0,1) \qquad (3.18)$$

ändert. $k_i$und z_j (Grundvarianz) sind Parameter, welche frei bestimmbar sind. Häufig wird $k_i = 1$ und $z_j = 0$ gesetzt um zu erreichen, dass die Mutation nur von der Fitness der Eltern beeinflusst wird (Gerdes, Klawonn und Kruse 2004, S. 125).

Um die im Vorgang verdoppelte Populationsgröße wieder auf ihre Ursprungsgröße reduzieren zu können, wird ein Selektionsoperator, ähnlich wie bei der (μ + λ)-ES, eingesetzt. Jedoch dient hier nicht der direkte Fitnesswert als Entscheidungsgrundlage, sondern es werden für jedes Chromosom aus der neuen Population zufällig q weitere Chromosomen

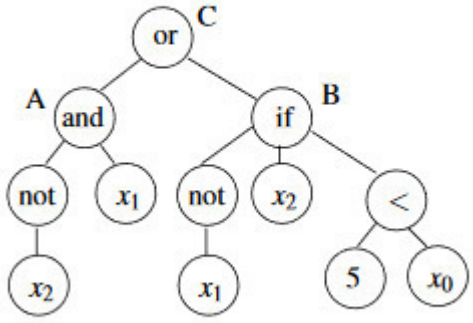
bestimmt, bei denen dann eine Wettkampfselektion durchgeführt wird. Bei dieser Form der Selektion wird die Anzahl der gewonnen direkten Vergleiche eines Chromosoms gezählt.

Daraufhin werden die q Chromosomen mit der besten Bewertung w in die Population $P(t+1)$ aufgenommen.

Genetische Programmierung

Die bisherigen Verfahren gingen stets davon aus, dass die Chromosomen eine feste Länge besitzen. Um Probleme wie beispielweise eine Flugroutenoptimierung mit variierender Anzahl von Zwischenknoten lösen zu können, müssen auch die Chromosomen eine flexible Länge besitzen (Gerdes, Klawonn und Kruse 2004, S. 127). In diesem Fall kommt die genetische Programmierung (GP) zum Einsatz.

Der Hauptoperator bei der GP ist die Rekombination, während die Mutation ähnlich wie bei dem GA nur eine untergeordnete Rolle spielt (Weicker 2015, S. 147). Bei dieser Betrachtung sind die Chromosomen zusammengesetzte Funktionen für die gilt: $C = F \cup T$. F ist eine endliche Menge aus Funktionen oder Funktionssymbolen (z.B. and, or, not, if, then, else,…) und T eine Konstante Menge von Terminalsymbolen (Konstanten, Variablen) (Gerdes, Klawonn und Kruse 2004, S. 127). Diese Struktur der Chromosomen kann als Programm interpretiert und durch Listen oder Syntaxbäume wie in **Abbildung 3-13** dargestellt werden (Gerdes, Klawonn und Kruse 2004, S. 127-129), (Weicker 2015, S. 147-148).



(a) Syntaxbaum

$5 < x_0$	x_1	x_2	A	B	C
false	*false*	*false*	*false*	*false*	*false*
false	*false*	*true*	*false*	*true*	*true*
false	*true*	*false*	*true*	*false*	*true*
false	*true*	*true*	*false*	*false*	*false*
true	*false*	*false*	*false*	*false*	*false*
true	*false*	*true*	*false*	*true*	*true*
true	*true*	*false*	*true*	*true*	*true*
true	*true*	*true*	*false*	*true*	*true*

(b) Wertetabelle

Abbildung 3-13: Beispielhaftes genetisches Programm als Syntaxbaum mit Wertetabelle (Weicker 2015, S. 148)

Jeder abgebildete Knoten enthält entweder ein Variablen- bzw. Konstantensymbol, welche die Blätter des Baums bilden, oder ein Funktionssymbol einer Funktion mit *n* Argumenten und dementsprechend *n* Nachfolgern. Möchte man die durch die Bäume repräsentieren Funktionen auswerten, beginnt man bei den Blättern und arbeitet sich von unten nach oben voran (Gerdes, Klawonn und Kruse 2004, S. 129).

Der prinzipielle Ablauf der GA ähnelt dabei den bisherigen Verfahren, wobei es jedoch Unterschiede beim Vorgehen in den einzelnen Schritten gibt. Zunächst werden geeignete Terminal- und Funktionssymbole festgelegt, welche zur Lösung des vorgegebenen Problems eingesetzt werden sollen. Desweiteren wird eine zufällige Startpopulation von Chromosomen (in Baumstruktur) erstellt. Um unnötig komplexe Baumstrukturen bei der Erzeugung der Chromosomen zu vermeiden, wird außerdem eine maximale Anfangstiefe *indepth* festgelegt (Gerdes, Klawonn und Kruse 2004, S. 130). Zur Bestimmung der Fitness werden *N* Fallbeispiele verwendet. Dabei werden zufällige Eingaben erzeugt und untersucht wie gut oder schlecht die jeweiligen Chromosomen (Bäume/Listen) das Problem lösen können. Mathematisch lässt ist die sogenannte *rohe Fitness* wie folgt berechnen:

$$r(l) = \sum_{i=1}^{N} err(l,i) \tag{3.19}$$

$err(l,i)$ ist dabei der Fehler der Liste l im i-ten Fallbeispiel (Gerdes, Klawonn und Kruse 2004, S. 131). Eine Alternative Fitnesswertbestimmung ist die *adjustierte Fitness*

$a(l) = \frac{1}{1+s(l)} \in [0{,}1\,]$, wobei $s(l) = 0$ eine optimale Liste l ist (3.20)

Zur Berechnung der Wahrscheinlichkeit für die Selektion wird die *normalisierte Fitness* verwendet, für die gilt (Gerdes, Klawonn und Kruse 2004, S. 132):

$$norm(l) = \frac{a(l)}{\sum_{i=1}^{s} a(l_i)} \in [0,1] \tag{3.21}$$

Der Hauptoperator bei der GP ist die Rekombination, bei der zufällig jeweils ein Knoten zweier Bäume ausgewählt und deren Unterbäume, wie in **Abbildung 3-14** zu sehen, miteinander vertauscht werden (Weicker 2015, S. 149), (Gerdes, Klawonn und Kruse 2004, S. 132).

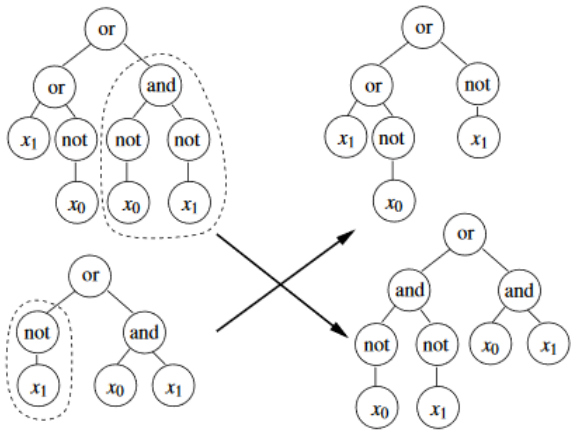


Abbildung 3-14: Baumtausch-Rekombination bei der GP (Weicker 2015, S. 150)

Die Besonderheit der genetischen Programmierung ist, dass selbst bei der Rekombination eines Chromosoms mit sich selbst zwei völlig neue Chromosomen entstehen können. Dementsprechend ist die Mutation bei diesem Verfahren nur ein optionaler Operator (Gerdes, Klawonn und Kruse 2004, S. 133).

Eine Mutation kann entweder eine zufällige Veränderung eines Knotens sein (Zufallsbaum-Mutation), oder es können zwei Unterbäume eines Chromosoms miteinander vertauscht werden (Baumtausch-Mutation), (Weicker 2015, S. 151-152).

Nachdem der Vorgang der Rekombination (und Mutation) abgeschlossen ist, beginnt eine weitere Iteration mit der Fitnessbewertung der neuen Population, sofern keine Abbruchbedingung erfüllt wurde.

3.4 Shuffled Frog Leaping Algorithmus

3.4.1 Ursprung in der Natur

Der naturanaloge Ausgangspunkt für die Idee des Shuffled Frog Leaping Algorithmus (SFLA) ist auf das mimetische Verhalten von Fröschen bei der Nahrungssuche zurückzuführen. Dabei agieren die Frösche in Gruppen, welche aus der Gesamtpopulation von Fröschen innerhalb eines Sumpfes gebildet werden. Mit dem Ziel das größtmögliche Nahrungsangebot zu lokalisieren, suchen die Frösche alle Steine, welche aus dem Sumpf herausragen ab, indem sie von Stein zu Stein springen (Eusuff und Lansey 2003, S. 212). Jeder Forsch besitzt dabei ein eigenes Bewusstsein und ist somit in der Lage eigene Entscheidungen zu treffen und Informationen weiterzugeben. Jede Gruppe von Fröschen führt zunächst auf Basis ihres "Gruppenbewusstseins" eine lokale Suche durch, indem gruppenintern Informationen ausgetauscht werden (Elbeltagi, Hegazy und Grierson 2005, S. 47). Nachdem die gruppeninterne Suche abgeschlossen ist, werden die Informationen global untereinander ausgetauscht, sodass sich die Frösche, aufgrund der erhaltenen Informationen, in Richtung Nahrungsquelle orientieren können.

3.4.2 Arbeitsweise

Diese Mischung aus lokaler Suche und globalem Informationsaustausch wird beim SFLA eingesetzt. Das Bewusstsein eines einzelnen Frosches wird als *Meme* bezeichnet, welches mehrere *Memotypes* enthält. Ähnlich wie das Gen eines Chromosoms repräsentiert ein Memotyp ein Charakteristikum des Memes. Die Gruppen von Fröschen werden als *Memeplexes* beschrieben, welche diskrete Variablen enthalten (Eusuff und Lansey 2003, S. 212). Analog zum Verhalten der Frösche beginnen die Memeplexes mit der lokalen Suche, welche solange durchgeführt wird, bis eine vorgegebene Anzahl von Schritten erreicht wurde und die Informationen global untereinander ausgetauscht werden können (Ming-Huwi 2011, S. 145). Auf Basis dieses Informationsaustausches werden die neuen, verbesserten Positionen im Suchraum (symbolisiert durch den Sumpf) bestimmt.

3.4.3 Mathematische Modelle

Der allgemeine Ablauf des SFLA, dessen einzelne Schritte im folgenden erläutert werden, ist in **Abbildung 3-15** dargestellt.

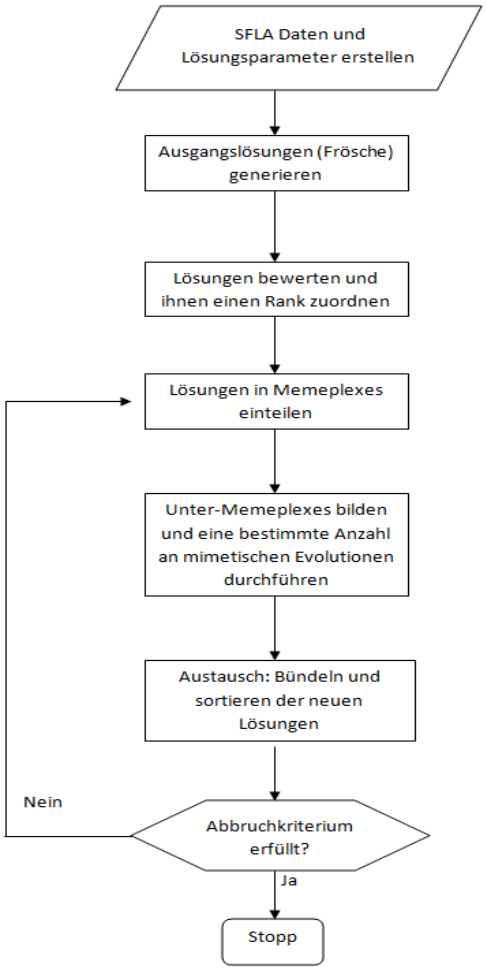


Abbildung 3-15: Ablaufdiagramm des SFLA in Anlehnung an (Eusuff und Lansey 2003, S. 213)

Der SFLA startet mit einer Ausgangspopulation P, welche aus zufällig generierten Fröschen X_i besteht. Die Güte der Lösungen (bzw. der Frösche) wird wie bei den Algorithmen zuvor durch einen Fitnesswert der Fitnessfunktion $f(X_i)$ definiert. Die Population aus Fröschen wird in m Memeplexes Y^k mit n Fröschen eingeteilt. Somit ergibt sich für die Population

$$P = m * n \tag{3.22}$$

Desweiteren werden die Frösche auf Grundlage ihres Fitnesswertes nach folgender Formel in absteigender Reihenfolge sortiert (Ming-Huwi 2011, S. 145):

$$Y^k = \{X_i^k = X_{k+m(i-1)}, i = 1, 2, \dots, n \; und \; k = 1, 2, \dots, m\} \tag{3.23}$$

Diese Reihenfolge ist entscheidend dafür, welcher Frosch welchem Memeplex zugeordnet wird. Der Frosch an Position eins wird dem ersten Memeplex zugeordnet, der Zweitplazierte dem zweiten Memeplex usw. bis dem m-ten Memeplex der Frosch an Position m

zugewiesen wird. Anschließend wird wieder mit dem ersten Memeplex begonnen, welches den Frosch an Position m+1 erhält. Dieser Vorgang wird solange wiederholt, bis alle Frösche einem Memeplex zugeordnet wurden (Elbeltagi, Hegazy und Grierson 2005, S. 47).

Darüber hinaus ist der Frosch mit dem besten und dem schlechtesten Fitnesswert eines jeden Memeplex bekannt. Diese Frösche werden mit X_b respektive X_w beschrieben. Die beste Fitness aller Frösche und somit das bisherige globale Optimum wird mit X_g deklariert (Ming-Huwi 2011, S. 145). Die lokale Suche wird von jeder Gruppe parallel durchgeführt und wird mathematisch wie folgt dargestellt:

$$D = r * (X_b - X_w) \tag{3.24}$$

$$X'_w = X_w + D, \quad mit\ D_{min} \leq D \leq D_{max} \tag{3.25}$$

D_{min} und D_{max} ist die minimale bzw. maximale Distanz, um die sich die Position eines Frosches verändert darf und r steht für eine Zufallszahl zwischen 0 und 1 (Ming-Huwi 2011, S. 145). In Kombination besagen diese beiden Formeln, dass der schlechteste Frosch X_w sich am besten Frosch X_b orientiert, um zur Futterstelle zu gelangen (Zhao und Lv 2015, S. 196). Es wird also stets nur die Position den schlechtesten Frosches verbessert. Geht aus diesem Vorgang ein besseres X'_w hervor, so ersetzt er den schlechtesten Frosch X_w. Sollte keine Verbesserung erzeugt werden, wird das lokale Optimum X_b durch den global besten Frosch X_g ersetzt und überprüft, ob diesmal eine Verbesserung erzielt wurde. Entsteht dabei immer noch keine Lösung, welche X_w ersetzen kann, so wird zufällig ein Frosch ausgewählt, welcher X_w ersetzt. Dieser Vorgang wird so lange wiederholt, bis eine vorgegebene Anzahl an Iterationen erreicht ist (Ming-Huwi 2011, S. 145).

Anschließend findet der globale Informationsaustausch statt. In dieser Phase tauschen die Frösche memeplexübergreifend Informationen miteinander aus, was dazu führt, dass die Memeplexes und die Frösche anhand ihres aktualisierten Fitnesswertes erneut in absteigender Reihenfolge sortiert werden. Zum Schluss werden die Frösche nach zuvor beschriebenem Muster den Memeplexes zugeordnet und eine neue Iteration gestartet, falls kein Abbruchkriterium erreicht wurde (Zhao und Lv 2015, S. 197).

Diese Grundversion des SFLA ist jedoch anfällig dafür in einem lokalem Optima zu stagnieren und konvergiert nur sehr langsam mit steigender Anzahl an Iterationen. Dies ist darauf zurückzuführen, dass sich der schlechteste Frosch am besten Frosch orientiert und in jeder Iteration annähert. Dadurch verkleinert sich der Bereich, in dem sich die Frösche bewegen können und die Vielfalt der Lösungen nimmt ab. (Zhao und Lv 2015, S. 197). Diesen Schwachstellen kann durch eine Verbesserung der lokalen Suche und des globalen Informationsaustausches, oder durch Kombination des SFLA mit anderen Algorithmen entgegengewirkt werden.

3.5 Harmony Search Algorithmus

3.5.1 Ursprung in der Natur

Die Analogie des Harmony Search (HS) Algorithmus entstammt, anders als bei den zuvor beschriebenen Algorithmen, nicht direkt aus der Natur, sondern basiert auf der Suche nach musikalischem Einklang bzw. Harmonie bei einer Jazz-Improvisation. Um eine musikalische Harmonie zu erreichen, müssen die Mitglieder der Band sich aufeinander abstimmen und in richtiger Reihenfolge die richtigen Töne spielen. Dabei sind die Musiker an sich, der Tonumfang der Instrumente, die Harmonie zu einem bestimmten Zeitpunkt, sowie das Publikum entscheidungsrelevante Parameter und beeinflussen die Suche nach der optimalen Reihenfolge (Geem 2010, S. 2-3). Bei einer Improvisation existieren drei Möglichkeiten wie diese durchgeführt wird (Geem 2010, S. 6): Durch

1) das Spielen von zufälligen Tönen eines Instruments,
2) das Spielen eines bekannten Liedes aus dem Gedächtnis heraus oder
3) das Spielen eines ähnlichen Liedes durch leichte Anpassung der Töne

Zu Beginn sind die Musiker noch nicht aufeinander abgestimmt und probieren verschiedene Kombinationen von Tönen aus, welche nicht im Einklang miteinander sind. Doch je mehr sie gemeinsam musizieren, desto besser harmonieren sie miteinander, bis sie schließlich eine harmonische Melodie erzeugen (Wang, Gao und Zenger 2015, S. 5-6).

3.5.2 Arbeitsweise

Durch die Überführung des Prozesses zur Suche nach musikalischer Harmonie in mathematische Gleichungen entstand der HS Algorithmus, welcher erstmals im Jahr 2000 von Zong Woo Geem zur Optimierung von Wasserverteilungssystemen vorgestellt wurde (Wang, Gao und Zenger 2015, S. 5) Wie die in Kapitel 3.5.1 erwähnten Parameter in den HS Algorithmus transformiert werden können ist in **Tabelle 3-3** aufgeführt.

Tabelle 3-3: Analoge Bedeutung von Parametern der Improvisation für den HS Algorithmus, in Anlehnung an (Geem, Kim und Loganathan 2001, S. 62)

Vergleichsfaktor	Harmonische Improvisation	Optimierung
Bester Zustand	Fantastische Harmonie	Globales Optimum
Bewertet von	Publikum (Ästhetischer Standard)	Zielfunktion
Bewertet mit	Tonhöhen von Instrumenten	Werte von Variablen
Prozessschritte	Jede Anwendung (von Tönen)	Jede Iteration

Auch beim HS Algorithmus werden die Lösungen (Töne) ausgehend ihrer möglichen Werte (Tonhöhen) iterationsweise verbessert, bis ein globales Optimum (harmonische Melodie) erreicht und die Zielfunktion minimiert bzw. maximiert (zufriedenes Publikum) wurde. Der Ablauf des HS Algorithmus ist in **Abbildung 3-16** dargestellt und wird im nächsten Unterkapitel unter Berücksichtigung mathematischer Gleichungen genauer erläutert.

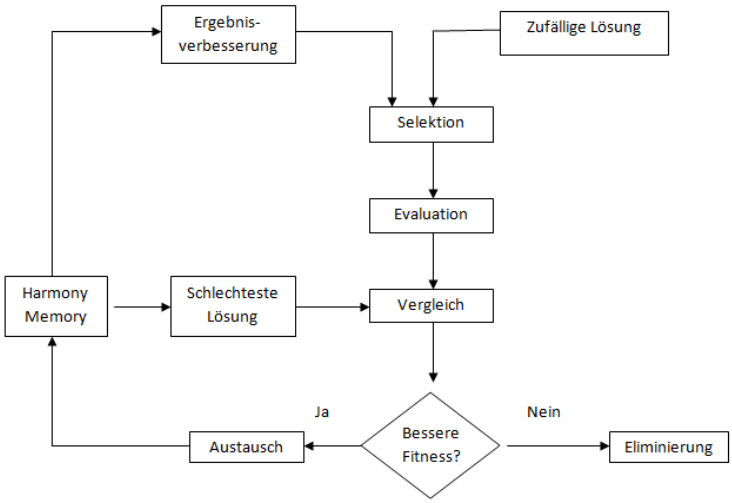


Abbildung 3-16: Ablaufdiagramm des HS Algorithmus in Anlehnung an (Wang, Gao und Zenger 2015, S. 7)

3.5.3 Mathematische Modelle

Der Ablauf des HS Algorithmus lässt sich in folgende vier Schritte einteilen (Geem, Kim und Loganathan 2001, 62):

1. Schritt: Initialisiere die "HS Memory" (HM)
   Die HM symbolisiert die zufällig gespielten Töne aus dem Gedächtnis eines Musikers. Überträgt man diese Logik nun auf den HS Algorithmus, so enthält die HM zufällig generierte Lösungen für ein Optimierungsproblem und bildet somit die Grundlage für die nachfolgenden Schritte. Eine HM mit M möglichen Lösungen und N Entscheidungsvariablen lässt sich mithilfe einer Matrix derartig darstellen (Bozorg-Haddad, Solgi und Loáiciga 2017, S. 126), (Geem 2010, S. 5):

$$HM = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_j \\ \vdots \\ X_M \end{bmatrix} = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_i^1 & \dots & x_N^1 \\ x_1^2 & x_2^2 & \dots & x_i^2 & \dots & x_N^2 \\ & & & \vdots & & \\ x_1^j & x_2^j & \dots & x_i^j & \dots & x_N^j \\ & & & \vdots & & \\ x_1^M & x_2^M & \dots & x_i^M & \dots & x_N^M \end{bmatrix} \tag{3.26}$$

   Hierbei ist X_j eine zulässige Lösung für das Problem, x_{ij} die dazugehörigen Entscheidungsvariablen und M die Kapazität der HM (Bozorg-Haddad, Solgi und Loáiciga 2017, S. 126).

2. Schritt: Improvisiere eine neue Harmonie aus HM
   In diesem Schritt wird auf Basis der HM eine neue Lösung $X^{Neu} = [x_1', x_2', \dots, x_N']$ erstellt. In Anlehnung an die in Kapitel 3.5.1 erwähnten Möglichkeiten, wie die Musiker bei einer Improvisation vorgehen können, existieren ebenfalls drei

Möglichkeiten nach denen eine neue Lösung Generiert werden kann (Geem 2010, S. 6-7):

- Memory Consideration (Berücksichtigung des Gedächtnis)
- Random Selection (Zufällige Selektion)
- Pitch Adjustment (Anpassung der Töne)

Memory Consideration

Hierbei werden nur die Entscheidungsvariablen zur Generierung einer neuen Lösung herangezogen, welche aus der zuvor erstellten HM stammen. Dabei werden zufällige Variablen x_i^j aus $HM = \{x_i^1, x_i^2, \dots, x_i^N\}$ ausgewählt und miteinander kombiniert, um eine neue Lösung zu entwickeln (siehe **Abbildung 3-17**) (Geem, Kim und Loganathan 2001, S. 7).

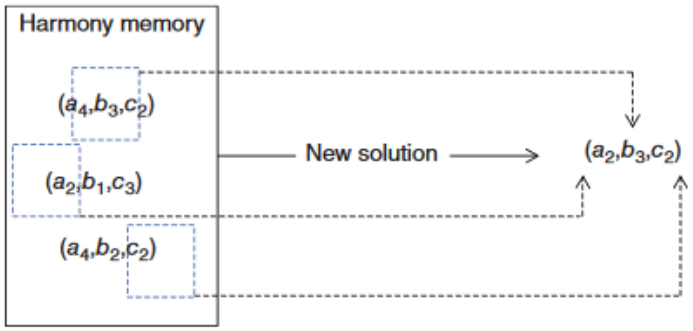


Abbildung 3-17: Generierung einer neuen Lösung auf Basis der "Memory Consideration" (Bozorg-Haddad, Solgi und Loáiciga 2017, S. 128)

Random Selection

Bei diesem Verfahren wird der komplette Wertebereich $\{x_i(1), \dots, x_i(k), \dots, x_i(K)\}$ bei diskreten Variablen, oder $x_i^{unten} \leq x_i \leq x_i^{oben}$ bei kontinuierlichen Variablen zur Erstellung einer neuen Lösung berücksichtigt (Geem 2010, S. 6). Schränkt man den Wertebereich für die Lösungen dahingehend ein, dass nur Entscheidungsvariablen aus der HM betrachtet werden, besteht die Gefahr, dass das globale Optimum nicht erreicht wird, falls eine Entscheidungsvariable der besten Lösung nicht Teil der HM ist. Aus diesem Grund ist es wichtig, stets mit einer kleinen Wahrscheinlichkeit Lösungen des kompletten Wertebereichs zuzulassen (Geem, Kim und Loganathan 2001, S. 63). Wie hoch diese Wahrscheinlichkeit ist, hängt von der "harmony memory considering rate" (HMCR) ab. Die HMCR ist ein Wert zwischen 0 und 1 und beschreibt die Wahrscheinlichkeit, mit der eine Entscheidungsvariable aus der HM als Teil einer neuen Lösung ausgewählt wird. Dementsprechend beträgt die Wahrscheinlichkeit 1-HMCR, dass zufällig eine Entscheidungsvariable aus dem kompletten Wertebereich ausgewählt wird, unabhängig davon, ob dieser aus der HM stammt oder nicht (Geem 2010, S. 4).

Pitch Adjustment

Mithilfe dieses Verfahrens können die neu generierten Lösungen verbessert und lokale Optima verlassen werden (Geem, Kim und Loganathan 2001, 63). Hierbei werden in Abhängigkeit der sogenannten "pitch adjusting rate" (PAR) einzelne Entscheidungsvariablen der Lösung verändert. Jedoch kann eine Variable nur den Wert ihrer direkten Nachbarn annehmen. Ist $x_i(k) = x_i^{neu}$ dann gilt für den veränderten (diskreten) Wert $x_i(k \pm 1)$. Für kontinuierliche variablen gilt analog $x_i^{neu} + \Delta$ (Geem 2010, S. 7). Besteht beispielweise der diskrete Wertebereich einer Variable aus {1, 3, 5, 6, 8, 9}, so kann {5} nur den Wert {3} oder {6} annehmen. Die Wahrscheinlichkeit, ob der kleinere oder größere Wert angenommen wird, ist dabei gleichverteilt, sodass bei einer PAR von 0,1 die Variable zu jeweils 5% den kleineren bzw. größeren Wert annimmt (Geem, Kim und Loganathan 2001, S. 63).

Die unten abgebildete Formel fasst alle drei Methoden zur Lösungsgenerierung mit ihren entsprechenden Einsatzwahrscheinlichkeiten zusammen:

$$x_i^{neu} \leftarrow \begin{cases} x_i \in HM = \{x_i^1, x_i^2, \dots, x_i^N & HMCR * (1 - PAR) \\ \begin{cases} x_i \in \{x_i(1), \dots, x_i(k), \dots, x_i(K)\} \\ \quad x_i \in [x_i^{unten}, x_i^{oben} \end{cases} & (1 - HMCR) \\ \begin{cases} x_i(k \pm 1) \quad wenn\, x_i(k) \in HM \\ \quad x_i + \Delta \quad wenn\, x_i \in HM \end{cases} & HMCR * PAR \end{cases} \tag{3.27}$$

3. Schritt: Memory Update
   Ist die neu generierte Lösung X^{neu} besser als die schlechteste Lösung X^{worst} aus HM, so wird X^{worst} eliminiert und X^{neu} nimmt ihren Platz in HM ein. Andernfalls wird X^{neu} eliminiert und nicht weitert betrachtet. Für den Fall, dass X^{neu} besser als alle Lösungen aus HM ist, kann ein Verfahren namens "accidentaling" angewendet werden. Dabei werden die Entscheidungsvariablen der besten Lösung X^{neu} ähnlich wie beim *Pitch Adjustment* verändert um möglicherweise eine noch bessere Lösung zu finden (Geem 2010, S. 9). Für die Entscheidungsvariablen der Lösung gilt dann:

$$x_i^{neu} \leftarrow \begin{cases} x_i(k \pm m) & für\ diskrete\ Variablen \\ x_i \pm \Delta & für\ kontinuierliche\ Variablen \end{cases} \tag{3.28}$$

4. Schritt: Abbruchkriterium überprüfen
   Sollte das Abbruchkriterium (z.B. max. Anzahl an Iterationen erreicht, oder globales Optimum gefunden) erreicht sein, endet der Algorithmus an dieser Stelle. Falls nicht, so wird beginnend mit dem zweiten Schritt eine neue Iteration gestartet (Geem 2010, S. 9).

4 Vergleich der genutzten Heuristiken

Nach der Erläuterung der Heuristiken in Kapitel 3, beinhaltet dieses Kapitel einen ausführlichen Vergleich zwischen diesen Verfahren. Auf Basis der zu lösenden Probleme wird insbesondere auf ihre Entscheidungsvariablen eingegangen, sowie die Vor- und Nachteile und eventuelle Verbesserungsmöglichkeiten der Verfahren erläutert. Für den Vergleich werden Ergebnisse anderer Autoren verwendet, welche die Algorithmen auf unterschiedliche Benchmarkprobleme anwandten.

4.1 Grundlegende Gemeinsamkeiten

Um nachvollziehen zu können, warum ein Vergleich zwischen diesen Algorithmen sinnvoll ist, werden zunächst die Gemeinsamkeiten der Algorithmen zusammenzufasst und hervorgehoben welche gemeinsame Basis sie besitzen.

Neben ihrer naturanalogen Grundlogik besitzen alle vorgestellten Algorithmen eine iterative Arbeitsweise. Schritt für Schritt werden auf Basis der vorherigen Iteration neue (bessere) Lösungen erzeugt, bis entweder das globale Optimum gefunden, oder ein Abbruchkriterium erfüllt wird. Diese Verfahren gehören zur Gruppe der Metaheuristiken, was bedeutet, dass sie auf ein breites Feld von Optimierungsproblemen angewendet werden können und sich nicht nur auf ein spezielles Problem beschränken. Desweiteren sind alle Verfahren populationsbasiert, da ihre Lösungen Teilmengen eines definierten Suchraumes sind und nicht nur eine Ausgangslösung betrachten (Domschke und Scholl 2010, S. 32). Alle Algorithmen sind von stochastischer Natur, da ihr Ablauf auf Wahrscheinlichkeiten und Zufallszahlen beruht. Um den stochastischen Einfluss mit in den Vergleich einfließen lassen zu können, werden Mittelwerte der erzielten Ergebnisse aus mehreren Anwendungen der Algorithmen als Vergleichskriterium herangezogen. Die Lösungen aller Algorithmen können als Vektoren, bzw. als 1 x N Matrizen der Form $X = (x_1, x_2, \dots, x_i, \dots, x_N)$ dargestellt werden. **Tabelle 4-1** fasst die Lösungen mit ihren Entscheidungsvariablen zusammen und beschreibt wodurch sie in den einzelnen Algorithmen symbolisiert werden.

Tabelle 4-1: Übersicht der Lösungen mit ihren Entscheidungsvariablen der einzelnen Algorithmen (eigene Darstellung)

	ACO	**PSO**	**EA**	**SFLA**	**HS**
Lösung Symbolisiert durch	Ameise	Partikel	Chromosom	Frosch	Harmonie
Entscheidungs-variable symbolisiert durch	Genutzte Kanten	Positionen der Partikel	Gene	Memotypes	Töne
Entscheidungs-variable	diskret	kontinuierlich	diskret oder kontinuierlich	diskret	diskret oder kontinuierlich

Von besonderen Interesse ist hierbei die Art der Entscheidungsvariablen. Ob diese diskret, kontinuierlich oder beides sein können, ist von höchster Relevanz für den Einsatz der Algorithmen zur Lösung von Optimierungsproblemen und bildet daher ein wichtiges Vergleichskriterium im folgendem Abschnitt. Eine genauere Erklärung der Entscheidungsvariablen und warum diese diskret bzw. kontinuierlich sind liefert Kapitel 4.2.

4.2 Vorauswahl der zu vergleichenden Heuristiken

Bevor die Algorithmen familienübergreifend miteinander verglichen werden, ist es sinnvoll eine Vorauswahl zu treffen. Dazu werden zunächst alle untersuchten Algorithmen innerhalb ihrer Familie miteinander Verglichen und eine Vorauswahl getroffen, um zu bestimmen welche Algorithmen für den abschließenden Vergleich in Frage kommen. Dabei dient ihre Lösungsgüte bei Anwendung auf verschiedene TSP als Vergleichsgrundlage. Die genutzten Probleminstanzen entstammen aus der Onlinebibliothek "Traveling Salesman Problem Library" (TSPLIB), in der unterschiedliche Problemstellungen von der Universität Heidelberg zusammengestellt wurden. Zu finden ist die Übersicht auf (Reinelt 1997). Die Zahlen im Namen der Probleme sind dabei stellvertretend für die Anzahl der Knoten bzw. Städte (z.B. Eil51 ≙ 51 Städte).

4.2.1 Ameisenalgorithmus

In Kapitel 3.1.3 wurden der Basisalgorithmus AS und seine beiden bedeutendsten Erweiterungen ACS und MMAS vorgestellt. Nun werden diese drei Algorithmen auf Grundlage der zu lösenden Probleme miteinander verglichen und ausgehend dieses Vergleichs ihre Vor- und Nachteile beschrieben.

Alle drei Algorithmen enthalten diskrete Entscheidungsvariablen, da die Kanten, welche die Knoten miteinander verbinden bekannt sind und somit aus einer abzählbaren, endlichen Menge ausgewählt und kombiniert werden können. Aus diesem Grund müssen keine weiteren Änderungen an den Algorithmen vorgenommen werden und können somit zur Lösung der vorliegenden Problemstellungen angewendet werden.

In **Tabelle 4-2** wird die Lösungsgüte der drei Algorithmen bei Anwendung auf TSP unterschiedlicher Größe verglichen. Um die Bedeutung des "trail-smoothing-Effekts" hervorzuheben, wird beim MMAS zwischen einem Verfahren *mit* und *ohne* "Pheromone Trail Smoothing" (PTS) unterschieden.

Tabelle 4-2: Vergleich zwischen Verfahren der ACO auf Basis der durchschnittlichen Ergebnisse bei der Lösung von TSP, in Anlehnung an (Stützle und Hoos 2000, S. 904)

Anzahl an Durchläufe: 25

Problem	Algorithmus	Durchschnitt	Optimum
eil51 (symmetrisch)	AS	437,3	426
	ACS	428,1	
	MMAS	427,6	
	MMAS+pts	**427,1**	
kroA100 (symmetrisch)	AS	22.471,4	21.282
	ACS	21.420,0	
	MMAS	21.320,3	
	MMAS+pts	**21.291,6**	
ry48p (asymmetrisch)	AS	15.296,4	14.442
	ACS	14.565,4	
	MMAS	14.553,2	
	MMAS+pts	**14.523,4**	
ftv170 (asymmetrisch)	AS	3154,5	2755
	ACS	2826,5	
	MMAS	2828,8	
	MMAS+pts	**2817,7**	

Aus **Tabelle 4-2** geht hervor, dass das MMAS+pts sowohl bei den symmetrischen als auch asymmetrischen TSP die beste Lösung findet. Dies ist darauf zurückzuführen, dass nur die beste Ameise die Pheromonkonzentration aktualisieren darf und somit der Fokus bei der Lösungssuche auf der momentan besten Tour liegt. Der trail-smoothing-Effekt" verhindert dazu eine Stagnation in einem lokalem Optimum, weshalb bei einer höheren Anzahl an Iterationen hierdurch das Ergebnis verbessert werden kann (vgl. Abschnitt 3.1.3). Jedoch besitzt das MMAS einen entscheidenden Laufzeitnachteil gegenüber den anderen ACO-Verfahren, welcher in **Abbildung 4-1** verdeutlicht wird. In diesem speziellem Fall ist ein Vergleich der Laufzeit repräsentativ durchführbar, da alle Verfahren in der gleichen Fallstudie unter denselben Bedingungen angewendet wurden.

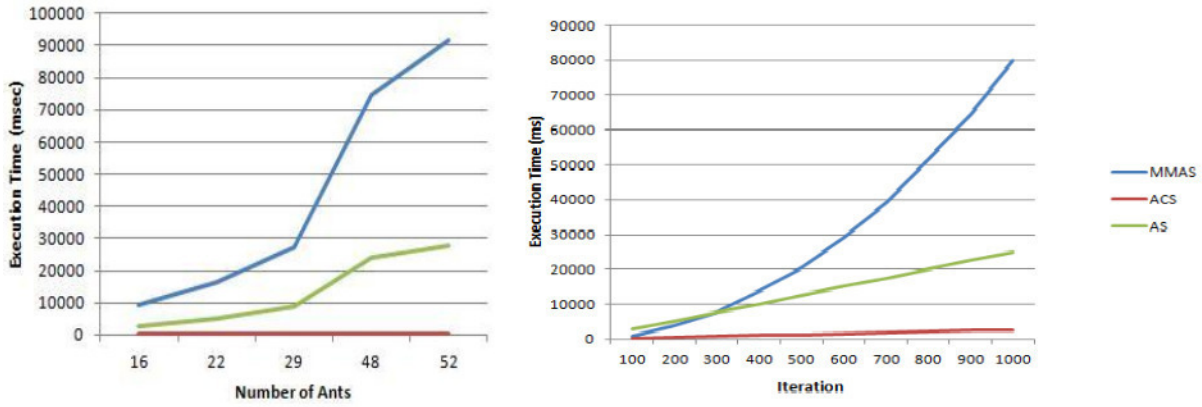


Abbildung 4-1: Laufzeiten der ACO-Verfahren in Abhängigkeit der Anzahl von Ameisen und Iterationen (Jangura und Kait 2017, S. 3)

Wie in **Abbildung 4-1** zu sehen ist, steigt die Laufzeit des MMAS um ein Vielfaches schneller als bei dem AS und ACS, je mehr Ameisen oder Iterationen verwendet werden. Dies liegt darin begründet, dass der MMAS-Algorithmus in jeder Iteration m Touren erstellt, welche selber eine Komplexität von n^2 aufweisen. Somit gilt für die Komplexität des ganzes Algorithmus $O(m * n^2)$. Üblicherweise wird in der Praxis m gleich der Anzahl der Knoten gewählt, sodass sich für den Algorithmus eine Komplexität von $O(n^3)$ ergibt, was zu der starken Laufzeiterhöhung führt (Stützle und Hoos 1997, S. 311).

Beim ACS dagegen wird der Lösungsraum großflächiger abgesucht, da mit einer gewissen Wahrscheinlichkeit (wenn $q > q_0$, vgl. Abschnitt 3.1.3) zufällige Knoten ausgewählt und somit neue Wege erforscht werden. Für den Fall $q \leq q_0$ beruht die Auswahl eines neuen Knotens auf einem Vergleich der Distanzen zum momentanen Standort, wodurch die kürzere Rechenzeit zu erklären ist. Jedoch wird durch die Vielfalt der Lösungen seltener das globale Optimum gefunden.

Die Laufzeit sowie die Lösungsgüte können jedoch durch Kandidatenlisten verbessert werden (vgl. Kapitel 3.1.3). **Tabelle 4-3** zeigt den Effekt verschiedener Kandidatenlisten am Beispiel des ASC zur Lösung des eil51 Problems.

Tabelle 4-3: Einfluss der Kandidatenliste auf die Lösungsgüte und Laufzeit, in Anlehnung an (Gambardella und Dorgio 1996, S. 5)

Größe der Kandidatenliste	Ø Lösung	Beste Lösung	Ø Zeit in Sek.
0	433,87	428	33,33
10	**431,00**	**426**	**13,93**
20	431,27	427	23,93
30	435,27	429	33,93
40	433,47	426	44,26
50	433,87	429	55,06

Anhand dieses Beispiels wird verdeutlicht, dass eine Kandidatenliste die Algorithmen stark verbessern kann, jedoch die Gefahr besteht, dass bei einer zu großen Kandidatenliste sowohl das Ergebnis, als auch die Laufzeit der Algorithmen verschlechtert wird.

Zusammenfassend lässt sich sagen, dass für kleinere Probleme mit wenigen Iterationen der Einsatz des MMAS vorteilhaft ist, aber aufgrund seiner langen Laufzeit das ACS bei größeren Problemen eingesetzt werden sollte.

Dennoch sind beide Algorithmen aufgrund ihrer unterschiedlichen Vorteile relevant für diese Arbeit und werden daher bei dem Vergleich mit den weiteren Algorithmen berücksichtigt.

4.2.2 Partikelschwarmoptimierung

Beim Vergleich der Entscheidungsvariablen in **Tabelle 4-1** miteinander fällt auf, dass die Entscheidungsvariablen der klassischen PSO kontinuierlich sind. Dies ist dadurch zu erklären, dass bei der herkömmlichen PSO die Partikel jede reellwertige Position im Suchraum annehmen können. Da die zu lösenden Probleme jedoch diskrete Entscheidungsvariablen enthalten, ist eine Diskretisierung der PSO notwendig. Zwei Verfahren der Diskretisierung werden exemplarisch vorgestellt. Eine Möglichkeit besteht darin, die Positionen der Partikel nach jeder Iteration zu runden, um so eine ganzzahlige Angabe der Positionen zu erhalten. Dieser Algorithmus nennt sich Integer PSO (IPSO) (Strasser, et al. 2016, S. 55). Einen anderen Ansatz nutzt die Binäre PSO (BPSO). Hierbei wird davon ausgegangen, dass die Positionen der Partikel als binäre Vektoren angegeben werden. Sind alle Bits auf 0, bewegt sich das Partikel nicht, wohingegen es die größte Distanz zurücklegt, wenn alle Bits den Wert 1 annehmen. Während die Formel (3.13) für das Geschwindigkeitsupdate nicht verändert wird, ändert sich die Berechnung für das Positionsupdate. Die Formel

$$S(v) = \frac{1}{1+\exp(-V_{i,j})} \tag{4.1}$$

bestimmt den Wert zwischen 0 und 1 für jeden Term der Geschwindigkeit, wobei $V_{i,j}$ der Wert der j-ten Variable von Partikel i ist. Anschließend wird eine Zufallszahl zwischen 1 und 0 erzeugt und mit $S(v)$ verglichen. Ist $S(v)$ größer als diese Zufallszahl, nimmt das Bit den Wert 1 an, ansonsten 0. Dadurch wird die neue Position des Partikels bestimmt (Strasser, et al. 2016, S. 56), (Kennedy und Eberhart 2002, S. 4105). Weitere Verfahren zur Diskretisierung sind zu finden in (Strasser, et al. 2016, S. 56) und (Hafiz und Abdennour 2016, S. 414).

Jedoch besitzen nicht alle Formen der diskreten PSO die gleiche Lösungsgüte. **Tabelle 4-4** vergleicht verschiedene diskrete PSO auf Basis unterschiedlicher Benchmarks des TSP. Um eine möglichst gute Übersicht zu erhalten, wurden als Vergleichsgrundlage TSP verschiedener Größen ausgewählt. Der Vergleich beinhaltet die reine diskrete PSO, sowie zusätzlich Weiterentwickelte Varianten dieses Verfahrens.

Tabelle 4-4: Gegenüberstellung verschiedener diskreter PSO beim TSP, in Anlehnung (Zhang, et al. 2007, S. 37), (Fan 2010, S. 3412) und (Cheng, et al. 2016, S. 181)

Anzahl an Durchläufe: 20

Problem	Algorithmus	beste Lösung	Durchschnitt	Optimum
burma14	DPSO (Clerk)	**30,8785**	31,61	30,8785
	PDPSO (Wang, Yang)	**30,8785**	**30,8785**	
	IHDPSO (Fan)	**30,8785**	**30,8785**	
	HDPSO (Ling-li, Zi-xing)	**30,8785**	**30,8785**	
Oliver30	DPSO (Clerk)	13.111	15.382	432,7406
	PDPSO (Wang, Yang)	423,741	**424,8267**	
	IHDPSO (Fan)	**423,7406**	426,1972	
	HDPSO (Ling-li, Zi-xing)	**423,7406**	429,3803	
eil51	DPSO (Clerk)	914,3	1025,2	426
	PDPSO (Wang, Yang)	436,773	440,781	
	IHDPSO (Fan)	**428,8718**	**430,4156**	
	HDPSO (Ling-li, Zi-xing)	429,9833	441,0769	

Aus **Tabelle 4-4** ist zu entnehmen, dass eine reine diskretisierte PSO nur für kleine Probleme das Optimum berechnen kann. Sobald das Problem aber komplexer wird, stößt die DPSO an ihre Grenzen und weitere Modifikationen sind notwendig, um Lösungen nahe des globalen Optimums zu erhalten. Doch auch innerhalb der modifizierten und diskreten PSO gibt es Unterschiede in der Lösungsgüte. Dies ist auf die verschiedenen Herangehensweisen bei der Diskretisierung und Modifizierung zurückzuführen. Wie gut oder wie schlecht die Lösungen der diskreten PSO im Vergleich zu den weiteren Heuristiken sind, wird in den nächsten Kapiteln untersucht.

4.2.3 Evolutionäre Algorithmen

Um eine Vorauswahl bei den evolutionären Algorithmen zu treffen, ist kein Vergleich ihrer Lösungsgüte für verschiedene Benchmarks notwendig. Dies kann alleine durch den Vergleich ihrer Arbeitsweise und Entscheidungsvariablen vonstattengehen. So sind die Entscheidungsvariablen der Evolutionsstrategien reellwertig (vgl. Abschnitt 3.3.3) und werden deshalb vornehmlich für kontinuierliche, ingenieurstechnische Herausforderungen verwendet. Das Konzept der evolutionären Programmierung wurde dagegen zur Reihenfolgeplanung endlicher Automaten entwickelt und wird deswegen nicht in den Vergleich mit einbezogen. Auch die genetische Programmierung wird im weiteren Verlauf

dieser Arbeit nicht näher betrachtet, da sie durch ihre Syntaxbaumstruktur zur Generierung von Computerprogrammen oder mathematischen Funktionen eingesetzt wird.

Aus diesem Grund wird in den folgenden Kapiteln ein besonderer Fokus auf den genetischen Algorithmus gelegt. Diese Wahl ist mit Blick auf seine Funktionsweise auch nachvollziehbar zu erklären. Die binäre Codierung der Gene und somit diskreten Entscheidungsvariablen, macht dieses Verfahren zu einem möglichen Lösungsansatz für diskrete Optimierungsprobleme. Auch die schnelle Verarbeitung durch die Computer, aufgrund der Codierung und die Tatsache, dass nur wenig Informationen über das Problem zur Lösung notwendig sind, machen den GA zu einem häufig angewendetem Verfahren in der kombinatorischen Optimierung.

4.2.4 Shuffled Frog Leaping Algorithmus

Beim SFLA ist keine Vorauswahl aus verschiedenen SFL-Verfahren zu treffen, da in dieser Arbeit der Basisalgorithmus im Vordergrund steht und somit keine unterschiedlichen Verfahren erläutert wurden (vgl. Abschnitt 3.4.3). Sollte beim abschließenden Vergleich dennoch ein modifizierter SFLA herangezogen werden, so wird dies erwähnt. Darum wird in diesem Kapitel hervorgehoben, warum der SFLA grundsätzlich zur Lösung der betrachteten Probleme herangezogen werden kann. Beim SFLA werden die Lösungen durch Memes bestehend aus diskreten Zufallsvariablen, den Memotypes, symbolisiert. Desweiteren besteht der Suchraum (symbolisiert durch einen Sumpf) aus abzählbar, endlichen Steinen mit bekannter Position. Somit ist dieses Verfahren auf diskrete Optimierungsprobleme anwendbar. Inwiefern sich der SFLA für die speziellen Problemstellungen eignet, wird im weiteren Verlauf der Arbeit untersucht.

4.2.5 Harmony Search Algorithmus

Für den HS-Algorithmus gilt prinzipiell dasselbe wie für den SFLA. Betrachtet wurde zunächst nur der Basisalgorithmus, weshalb kurz seine (theoretische) Tauglichkeit zur Lösung der genannten Probleme überprüft wird. Die Lösung wird hierbei von einer Harmonie bestehend aus einer Menge von bekannten, abzählbar endlichen Tönen symbolisiert. Die spielbaren Töne sind dabei diskrete Werte aus einem gegebenen Tonumfang. Das besondere am HS-Algorithmus ist, dass die Entscheidungsvariablen auch kontinuierlich sein können ohne, dass der Algorithmus unpräziser wird (Moh'd Alia und Mandava 2011, S. 55). Diese Information dient aber nur der Vollständigkeit halber und es wird im weiteren Verlauf nur der diskrete HS-Algorithmus betrachtet.

4.3 Gegenüberstellung der Algorithmen

Nachdem eine Vorauswahl an zu vergleichenden Algorithmen durchgeführt und ihre Entscheidungsvariablen untersucht wurden, kann nun ihre Tauglichkeit auf spezifische Problemstellungen in logistischen Netzwerken anhand eines Vergleichs überprüft werden. Da aufgrund der unterschiedlichen Leistungsstärke der verwendeten Computer und verschiedenen Rahmenbedingungen der Fallstudien ein Vergleich hinsichtlich der Laufzeit nicht repräsentativ durchführbar ist, wird in dieser Arbeit auf einen Laufzeitvergleich verzichtet und die Lösungsgüte der Algorithmen als Hauptvergleichsparameter ausgewählt.

4.3.1 Traveling Salesman Problem als Benchmark

Als erste Vergleichsgrundlage werden verschiedene bekannte Benchmarks des TSP verwendet. Im Zuge dieses Vergleichs wird das Verhalten der Algorithmen bei Lösung von symmetrischen und asymmetrischen TSP verschiedener Größen untersucht. Der Vergleich beruht auf den Ergebnissen unterschiedlicher Autoren, welche im Rahmen von Fallstudien die genannten Algorithmen auf die verschiedenen Problemstellungen anwendeten. Das Parametersetting für alle Fallstudien ist im *Anhang B* gegeben.

Lösungsgüte

Mit den Werten für die Parameter aus Anhang B wurden die verschiedenen Algorithmen zu Lösung der beschriebenen Probleme in mehreren unabhängigen Durchgängen angewandt und ihre Ergebnisse notiert. Auf dieser Grundlage wird nun zunächst die Lösungsgüte der Algorithmen gegenübergestellt. Dazu sind in **Tabelle 4-5** die jeweils beste Lösung und der Mittelwert aller Durchgänge dargestellt und mit der besten bekannten Lösung des Problems verglichen. Desweiteren wird eine Spalte "% Error" erstellt, welche die Lösungsgüte der Algorithmen bestimmt. Je nachdem ob ein Mittelwert der Ergebnisse oder das beste erzielte Ergebnis eingesetzt wird, wird eine der folgenden Formeln verwendet:

$$\%\, Error_{avg} = \frac{\text{Ø Lösung} - \text{Optimale Lösung}}{\text{Optimale Lösung}} * 100 \qquad (4.2)$$

$$\%\, Error_{best} = \frac{\text{beste Lösung} - \text{Optimale Lösung}}{\text{Optimale Lösung}} * 100 \qquad (4.3)$$

Bei dieser Kontrollgröße ist ein niedriger Wert erstrebenwert, da er die Abweichung der Lösungen zum Optimum beschreibt. Um den stochastischen Einfluss der Algorithmen zu berücksichtigen, wird sofern möglich, Formel (4.2) in dieser Arbeit bevorzugt angewendet.

Tabelle 4-5: Vergleich von Lösungen verschiedener Algorithmen beim TSP (eigene Darstellung)

Problemstellung	Algorithmus	Durchläufe	Beste Lösung	Durchschnitt	% $Error_{avg}$	Optimum
Burma14 (symmetrisch)	MMAS	-	-	-	-	30,8785
	ACS	-	-	-	-	
	CPSO	100	30,9062	30,9245	0,15	
	GA	-	-	-	-	
	SFLA	50	**30,8785**	**30,8785**	**0,00**	
	HS	-	-	-	-	
Eil51 (symmetrisch)	MMAS	25	**426**	427,2	0,28	426
	ACS	15	**426**	428,06	0,48	
	CPSO	100	427	442	3,76	
	GA	20	**426**	**426**	**0,00**	
	SFLA	50	428,87	436,76	2,53	
	HS	10	**426**	426,3	0,07	
KroA100 (symmetrisch)	MMAS	25	**21.282**	21.352,05	0,33	21.282
	ACS	15	**21.282**	21.420	0,65	
	CPSO	-	-	-	-	
	GA	20	**21.282**	**21.282**	**0,00**	
	SFLA	-	-	-	-	
	HS	10	**21.282**	**21.282**	**0,00**	

In dieser Tabelle sind die Ergebnisse von TSP verschiedener Größen gegenübergestellt wobei die besten Ergebnisse hervorgehoben sind. Dabei bedeuten die leeren Felder, dass das Problem in den betrachteten Fallstudien nicht thematisiert wurde. Es ist zu sehen, dass der GA bei beiden betrachteten Problemen in jedem Durchlauf die optimale Lösung liefert. Doch auch der HS-Algorithmus erreicht bei KroA100 stets das globale Optimum und weicht bei Eil51 durchschnittlich nur um 0,07% ab. Desweiteren liefern beide ACO-Algorithmen annähernd optimale Lösungen mit einer durchschnittlichen Abweichung von unter einem Prozent. Am schwächsten schnitten der SFLA und die Chaotische PSO (CPSO), ein weiteres Verfahren der diskreten PSO, ab. Insbesondere bei der CPSO ist dies damit zu erklären, dass die ursprüngliche PSO zur Lösung kontinuierlicher Probleme entwickelt wurde (vgl. Abschnitt 3.2.3 und 4.3.2) und durch Modifikationen an diskrete Problem angepasst wurde. Aus diesem Grund liefern diskrete PSO in der Regel schlechtere Resultate, als Algorithmen welche direkt zur Lösung diskreter Problemstellungen entwickelt wurden. Nichtsdestotrotz liefert auch die CPSO für kleinere Probleme gute Lösungen.

Verhalten bei wachsender Problemstellung

Wie genau sich die Algorithmen bei wachsender Problemstellung durch Erhöhung der Knotenanzahl verhalten, wird im diesem Abschnitt untersucht. Dazu werden die Abweichungen der besten berechneten Lösungen vom Optimum, in Abhängigkeit der Knotenanzahl gegenübergestellt (siehe **Abbildung 4-2**). Um den Vergleich so repräsentativ wie möglich zu gestalten und Verfälschungen der Ergebnisse entgegenzuwirken, wird sich

pro Algorithmus auf eine Fallstudie beschränkt, sodass innerhalb eines Algorithmus die gleichen Bedingungen und Parametersettings gelten. Um möglichst viele Probleminstanzen verschiedener Größen abdecken zu können, mussten beim SFLA jedoch zwei unterschiedliche Fallbeispiele herangezogen werden. Modifikationen am Basisalgorithmus sind durch kurze Erklärungen der durchgeführten Verbesserungen gekennzeichnet. Der Vergleich beruht auf den Ergebnissen symmetrischer TSP von:

- **MMAS:** (Stützle und Hoos 1997, S. 311)
  - MMAS mit Kandidatenlisten.
- **ACS**: (Gambardella und Dorgio 1996, S. 4)
  - Herkömmliches ACS (vgl. Abschnitt 3.1.4)
- **DPSO**: (Shi, et al. 2007, S. 173)
  - Binäre PSO bei der zur Erhöhung der Konvergenzgeschwindigkeit durch einen sogenannten "*delete-crossover Process*" verhindert wird, dass sich die Flugbahnen zweier Partikel kreuzen.
- **GA**: (Freisleben und Merz 1996, S. 620)
  - Kombination von GA und lokaler Suche.
- **SFLA**: (Luo, Yang und Li 2008, S. 231) und (Zhang und Wu 2012, S. 94-95)
- **HS**: (Bouzidi und Riffi 2014, S. 158)
  - Herkömmlicher HS Algorithmus (vgl. Abschnitt 3.5.3)

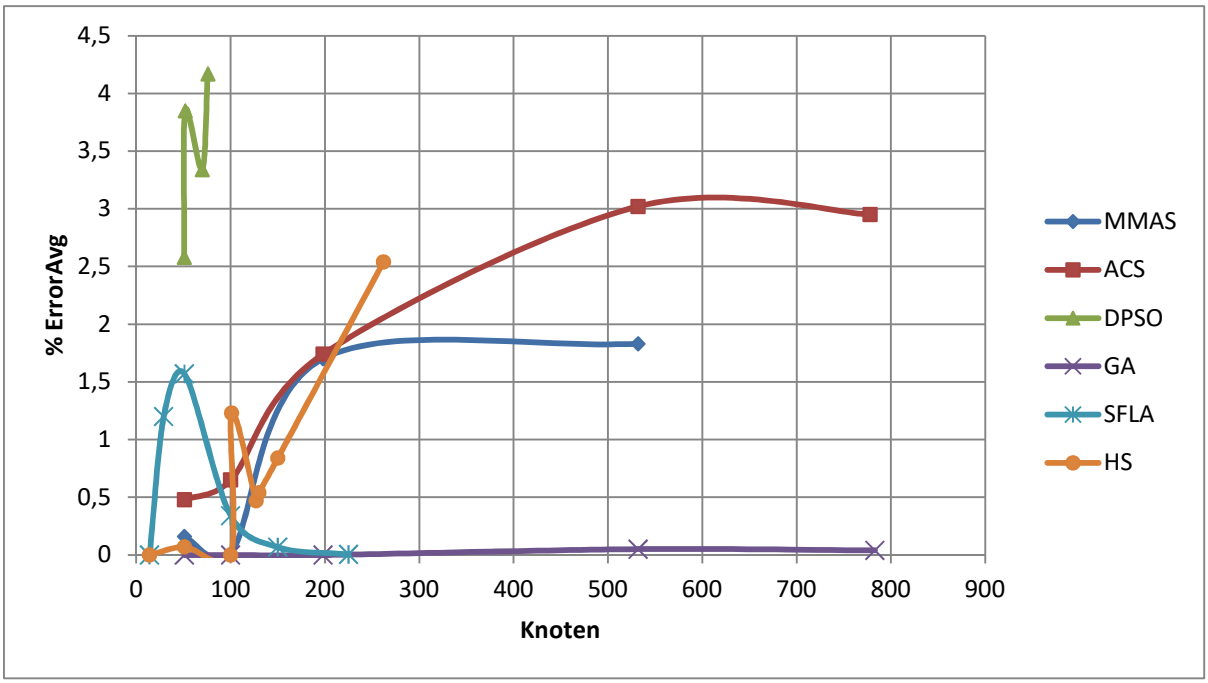


Abbildung 4-2: Abweichung vom Optimum in Abhängigkeit der Knotenanzahl bei STSP (Eigene Darstellung)

Die Abbildung verdeutlicht, dass die DPSO die höchste Abweichung zum Optimum aufweist. Sogar für kleinere bis mittlere Probleme, mit weniger als 100 Knoten, überteigt sie die Abweichung aller anderen Algorithmen, selbst wenn diese für größere TSP eingesetzt wurden. Der starke Abfall der Abweichung des SFLA ist auf die Nutzung der beiden eingangs

erwähnten Fallbeispiele zurückzuführen. Ab einer Knotenanzahl von n=51 kam der verbesserte Algorithmus, "Improved SFLA" zum Einsatz. Bei diesem Verfahren wird nicht nur der schlechteste Frosch verbessert, sondern die schlechtesten 50% der kompletten Population (Zhang und Wu 2012, S. 93). Dadurch lässt sich erkennen, dass Algorithmen im Laufe der Zeit stets verbessert werden können und müssen, um der steigenden Komplexität der Problemstellungen entgegenwirken zu können. Die Kurvenverläufe des MMAS und ACS decken sich mit den Folgerungen aus Abschnitt 4.2.1, in dem beobachtet wurde, dass das MMAS in der Regel bessere Ergebnisse als das ACS liefert. Auch der HS Algorithmus liefert gute Ergebnisse, wird mit steigender Knotenanzahl jedoch ungenauer. Wie auch beim ersten Vergleich schneidet der GA am besten ab. Selbst bei großen Problemstellungen mit bis zu 783 Knoten liefert der GA Ergebnisse, welche sich nur um maximal 0,05% vom Optimum unterscheiden.

Asymmetrische TSP

Inwiefern sich die Lösungen ändern, wenn $d_{ij} \neq d_{ji}$ gilt, wodurch das TSP asymmetrisch ist, wird im folgenden Abschnitt untersucht. Es existieren keine verwertbaren Daten des SFLA zur Anwendung auf asymmetrische TSP, weshalb der Algorithmus bei diesem Vergleich fehlt. Der Vergleich in **Tabelle 4-6** beruht auf den Ergebnissen von:

- **MMAS** und **ACS**: (Stützle und Hoos 1997, S. 311)
  - MMAS mit Kandidatenlisten und herkömmliches ACS (vgl. Abschnitt 3.1.4)
- **HDPSO**: (Li, et al. 2006, S. 187)
  - Kombination aus diskreter PSO und lokaler Suche. Die Position eines Partikels ist durch eine Matrix gegeben, welche die Position einer Stadt in einer Tour kennzeichnet. Die Geschwindigkeit beschreibt hierbei die Neigung eines Partikels eine Stadt als einen Vorgänger für eine andere Stadt zu wählen.
- **GA**: (Choi, Kim und Kim 2003, S. 782)
  - Hierbei werden absichtlich unzulässige Lösungen nahe des Optimums mit einbezogen, um daraus anschließend mithilfe eines sogenannten "patching Algorithmus" gute zulässige Lösungen zu generieren.
- **HS**: (Boryczka und Szwarc 2018, S. 9)
  - Herkömmlicher HS Algorithmus (vgl. Abschnitt 3.5.3)

Tabelle 4-6: Vergleich zwischen Algorithmen bei asymmetrischen TSP (Eigene Darstellung)

Problem	Algorithmus	Durchläufe	Beste Lösung	Ø Lösung	% $Error_{avg}$	Optimum
ry48p	MMAS	25	**14.422**	**14.465**	**0,30**	14.422
	ACS	15	**14.422**	14.565,45	0,99	
	HDPSO	10	-	14.675,30	1,76	
	GA	3	14.422	14.470	0,33	
	HS	10 Min.[7]	14.459	14.522,07	0,69	
ft70	MMAS	25	**38.690**	38.913,50	0,62	38.673
	ACS	15	38.781	39.099,05	1,10	
	HDPSO	10	-	39.137,40	1,20	
	GA	3	38.707	**38.755,70**	**0,21**	
	HS	10 Min.	39.427	40.146,03	3,81	
krol124	MMAS	25	36.416	36.572,85	0,95	36.230
	ACS	15	**36.241**	36.857	1,73	
	HDPSO	10	-	37.513,20	3,54	
	GA	3	36.393	**36.420**	**0,52**	
	HS	10 Min.	39.171,53	38.000	4,89	
ftv170	MMAS	25	2787	2807,75	1,91	2755
	ACS	15	2774	2826,47	2,59	
	HDPSO	10	-	2959,4	7,42	
	GA	3	**2761**	**2761**	**0,22**	
	HS	10 Min.	3101	3333,6	21,00	

Auch aus diesem Vergleicht geht hervor, dass der GA und das MMAS im Durchschnitt die besten Ergebnisse liefern. Dabei ist insbesondere der GA hervorzuheben, da seine durchschnittlichen Ergebnisse auch bei größeren Problemen mit mehr als 100 Knoten nur geringfügig vom globalen Optimum abweichen, wobei anzumerken ist, dass schon ab einer Knotenanzahl von $n=70$ kein Algorithmus das globale Optimum erreicht. Die hybride diskrete PSO (HDPSO) ist eine Kombination aus der DPSO und lokaler Suche. Aufgrund seiner Komplexität von $O(n^3)$ muss eine Nachbarschaft der Länge k definiert werden (vgl. Abschnitt 3.2.2), um der langen Laufzeit bei größeren TSP entgegenzuwirken. Allerdings wird dieses Verfahren mit zunehmender Problemgröße ungenauer und liefert schlechtere Ergebnisse als der GA und die beiden ACO-Verfahren. Ein Grund für das schlechte Abschneiden des HS-Algorithmus könnten die Rahmenbedingungen bei der Durchführung der Fallstudie sein. Es gab keine vordefinierte Anzahl an Durchläufen, sondern es wurden die Ergebnisse nach einer Rechenzeit von 10 Minuten verwendet, weshalb der Vergleich des HS-Algorithmus nur bedingt repräsentativ ist. Allerding ist zu erkennen, dass sich die Ergebnisse mit steigender Knotenanzahl immer weiter vom globalem Optimum entfernen. Um die Effizienz der Algorithmen bei wachsender Problemgröße besser zu illustrieren, werden in **Abbildung 4-3** Werte aus der Spalte "% $Error_{avg}$" gegenübergestellt.

[7] Die Ergebnisse dieser Fallstudie wurden nicht anhand von mehreren Durchläufen erreicht, sondern es wurden die Ergebnisse, welche nach einer Laufzeit von 10 Minuten erreicht wurden, verwendet.

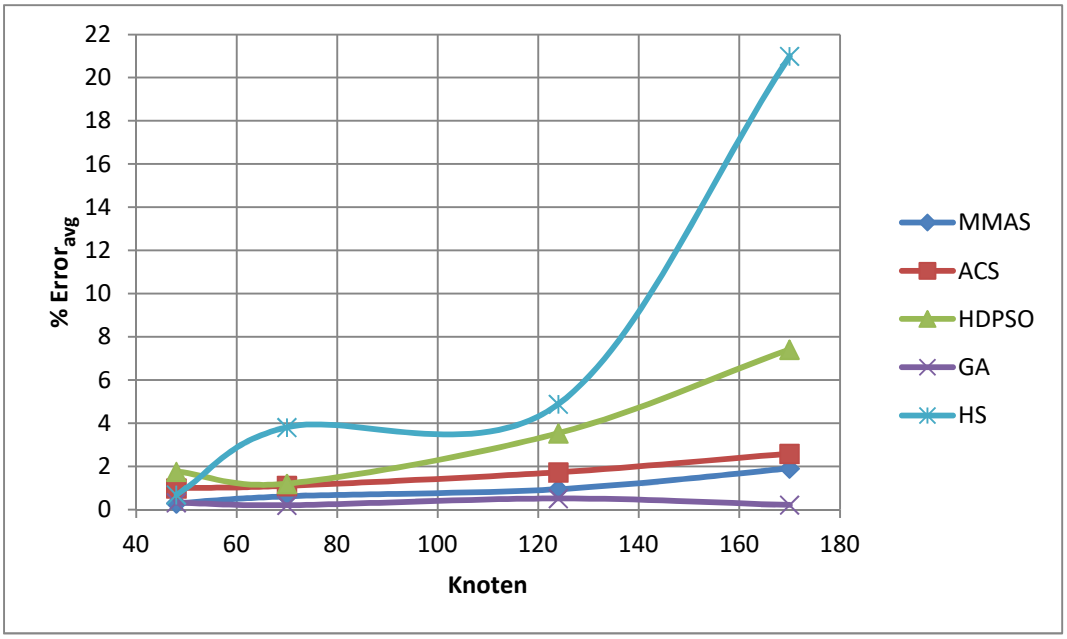


Abbildung 4-3: Abweichung vom Optimum in Abhängigkeit der Knotenanzahl bei ATSP(Eigene Darstellung)

Aus dieser Abbildung ist zu entnehmen, dass der GA am stabilsten ist und auch bei größeren Problemstellungen gute Ergebnisse liefert. Doch auch das MMAS und ACS weisen nur leichte Erhöhungen der Abweichungen auf und erreichen bei wachsender Knotenanzahl Ergebnisse nahe des globalen Optimums. Dahingegen ist beim HS Algorithmus ein starker Anstieg der Abweichung zu erkennen, je größer das Problem wird. Somit ist seine Anwendung nur für kleine Probleme mit $n \leq 50$ Knoten zu empfehlen.

4.3.2 Vehicle Routing Problem als Benchmark

In diesem Abschnitt werden die Algorithmen hinsichtlich ihrer Anwendung zur Lösung des Vehicle Routing Problems verglichen. Die grundsätzliche Ausgangslage ist die Selbe wie beim TSP, jedoch müssen von nun an mehr Nebenbedingungen berücksichtigt werden, welche ein jedes VRP definieren. Aufgrund der vielen unterschiedlichen Arten des VRP (vgl. Abschnitt 2.3.3) beschränkt sich diese Arbeit auf die in der Literatur am häufigsten untersuchen Problemstellungen im Zusammenhang mit den genannten Algorithmen. Diese Probleme sind das CVRP und das VRPTW. Zudem wird das MDVRP betrachtet und herausgearbeitet inwieweit die Belieferung ausgehend von mehreren Depots Einfluss auf die Lösungsgüte hat. Allerdings wurden in der Literatur nicht alle Algorithmen hinsichtlich dieser drei Probleme untersucht, weshalb vereinzelte Algorithmen nicht Bestandteil aller Vergleiche sind. Es wird allerdings jeder Algorithmus auf mindestens eine der Problemstellungen angewendet. Wegen der vorteilhafteren Anwendung des ACS gegenüber dem MMAS bei größeren Problemen, beschränken sich die folgenenden Vergleiche auf das ACS. Die verwendeten Benchmarks von CVRP, VRPTW und MDVRP entstammen dabei erneut aus Onlinebibliotheken und sind zu finden unter (Xavier 2014), (Solomon 2005) bzw. (Díaz 2006).

Capacitated Vehicle Routing Problem

Als Vergleichsgrundlage dienen die Benchmarks von Christofides, Mingozzi und Toth. **Tabelle 4-7** fasst alle relevanten Daten der Probleminstanzen zusammen. Dabei sind die Kunden der Instanzen C11-C14 in Clustern gruppiert. Eine beispielhafte Kundenverteilung ohne Cluster und mit einer solchen Gruppierung ist in **Abbildung 4-4** gegenübergestellt.

Tabelle 4-7: Überblick über die Parameter der Benchmarks von Christofides, Mingozzi und Toth, in Anlehnung an (Xavier 2014)

Instanz	Anzahl Kunden	Anzahl Fahrzeuge	Fahrzeug-kapazität	Beste bekannte Lösung
C1	50	5	160	524,61
C2	75	10	140	835,26
C3	100	8	200	826,14
C4	150	12	200	1028,42
C5	199	17	200	1291,29
C6	50	6	200	555,43
C7	75	11	140	909,68
C8	100	9	200	865,95
C9	150	14	200	1162,54
C10	199	18	200	1395,85
C11	120	7	200	1042,1
C12	100	10	200	819,56
C13	120	11	200	1541,14
C14	100	11	200	866,37

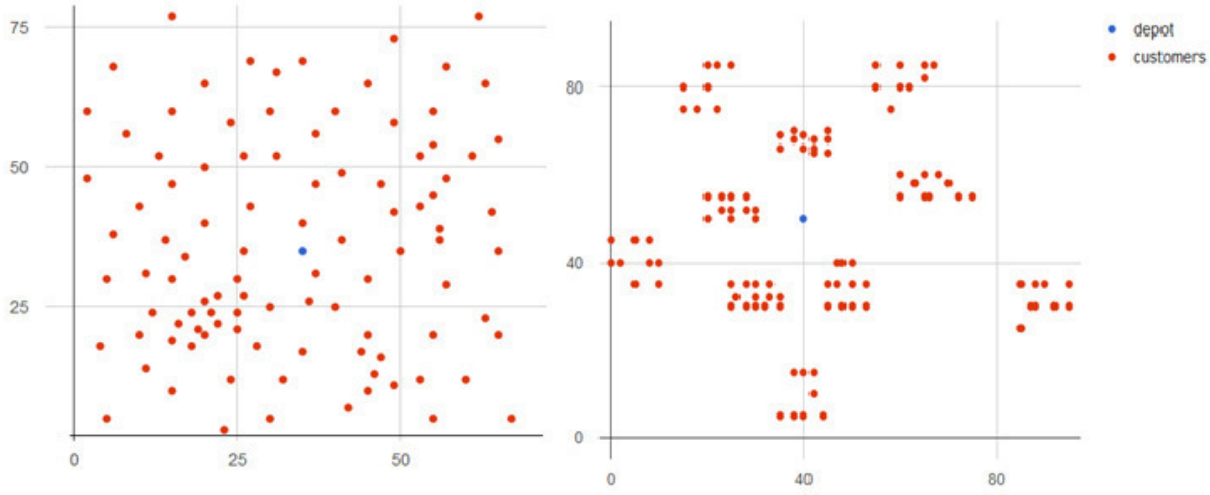


Abbildung 4-4: Kundenverteilung ohne Cluster (links) und mit (rechts) (Xavier 2014)

Während im rechten Teil der **Abbildung 4-4** die Kunden geographisch zu Gruppen angeordnet sind, verteilen sich die Kunden auf der linken Seite unregelmäßig auf dem kompletten Graphen. Ob und inwiefern dies Auswirkungen auf die Lösungsgüte der Algorithmen hat, wird im Folgenden ebenfalls untersucht.

Die Gegenüberstellung der Ergebnisse und Laufzeiten in **Tabelle 4-8** beruht auf den Ausarbeitungen folgender Autoren:

- **IACS**: (Chen und Ting 2006, S. 122)
  - Bei dem "*improved ACS*" wird die lokale Pheromon-Update-Regel modifiziert indem vor dem Pheromon Update eine lokale Suche auf die s besten Ameisen angewendet wird.
- **HybPSO**: (Marinakis, Marinaki und Dounias 2010, S. 468)
  - Die hybride PSO ist eine Kombination aus einer diskreten PSO, der "*Multiple phase neighborhood Search*", der "*expanding neighborhood search*" und der "*path linking strategy*".
- **GA**: (Prins 2004, S. 1997)
  - Kombination des GA mit der lokalen Suche.

Tabelle 4-8: Vergleich verschiedener Algorithmen zur Lösung des CVRP (Eigene Darstellung)

	ACS	**PSO**	**GA**
Instanz	Beste Lösung	Beste Lösung	Beste Lösung
C1	524,61	524,61	524,61
C2	835,32	835,26	835,26
C3	831,2	826,14	826,14
C4	1037,02	1029,54	1030,46
C5	1320,5	1294,13	1296,39
C6	555,43	55,43	555,43
C7	909,68	909,68	909,86
C8	865,94	868,45	865,94
C9	1169,69	1164,35	1162,55
C10	1409,61	1396,18	1402,75
C11	1042,1	1044,03	1042,11
C12	831,83	819,56	816,56
C13	1545,68	1544,18	1542,86
C14	866,37	866,37	866,37

Hieraus ist zu erkennen, dass für kleinere Probleminstanzen (C1, C2, C6 und C7) mit weniger als 100 Kunden, mit einer Ausnahme (IACS für C2) alle Algorithmen in der Lage sind die optimale Tour zu berechnen. Auch für mittlere und größere Probleme sind die Abweichungen der besten berechneten Touren vom Optimum minimal, wie **Abbildung 4-5** bestätigt.

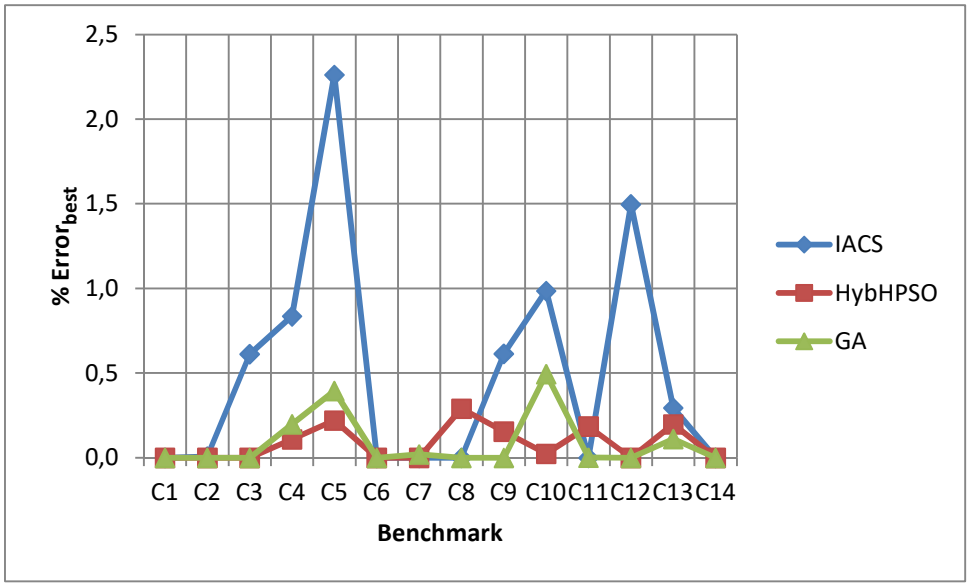


Abbildung 4-5: Abweichungen der besten berechneten Lösungen vom Optimum (Eigene Darstellung)

Es wird deutlich, dass das IACS bei größeren Problemen, insbesondere bei den Instanzen C5, C10 und C12 an seine Grenzen stößt und eine Abweichung von bis zu 2,3% aufweist. Im Gegensatz dazu bleibt die Abweichung der anderen beiden Algorithmen konstant unter 0,5%. Im Hinblick auf die Anordnung der Kunden im Graphen sind keine Besonderheiten in Bezug auf die Lösungsgüte festzustellen. Die gute Performance des HybPSO zeigt, dass ,entgegen der Annahmen aus Abschnitt 4.3.1, sich die diskrete PSO zur Lösung kombinatorischer Optimierungsprobleme eignet. ihre Lösungsgüte aber stark von der Art der Diskretisierung und Modifizierung abhängig ist. Es sei dabei angemerkt, dass die Unterschiede zwischen den Algorithmen marginal ausfallen und auch eine Abweichung von 2,3% ein gutes Ergebnis für große Probleminstanzen ist.

Vehicle Routing Problem with Time Windows

Das nächste betrachtete Problem bei Tourenplanungen ist das VRPTW. Beim VRPTW müssen neben allen Nebenbedingungen des CVRP zusätzlich Zeitfenster für die Belieferung der Kunden berücksichtigt werden. Neben dieser zusätzlichen Restriktion, steht in diesem Vergleich die Minimierung zweier Zielfunktionen im Vordergrund. Das Primärziel ist eine Minimierung der Touren bzw. der eingesetzten Fahrzeuge. Da jedes Fahrzeug nur eine Tour fährt, ist die Anzahl der Fahrzeuge identisch mit der Anzahl der Touren. Als zweites Ziel muss mit für die minimale Anzahl von Fahrzeugen eine optimale Reihenfolge der Kundenbelieferung erstellt werden, um die Transportwege zu minimieren.

Betrachtet werden 56 Benchmarks des VRPTW von (Solomon 2005), welche in sechs Problemtypen (C1, C2, R1, R2, RC1, RC2) mit jeweils acht bis zwölf Benchmarks eingeteilt wurden. Die Probleme der Klasse C enthalten eine gruppierte Kundenverteilung, dessen Zeitfenster auf einer bekannten Lösung beruhen. Bei der Problemklasse R sind die Kunden zufällig über den kompletten Graphen verteilt. Die Klasse RC beinhaltet eine Kombination aus gruppierten und zufällig verteilten Kunden. Zudem erfolgt eine Einteilung nach der Größe der Zeitfenster und der Ladekapazität der Fahrzeuge. Bei der Problemklasse 1 sind

beide Parameter klein gewählt worden, wohingegen bei der Klasse 2 große Zeitfenster und hohe Kapazitäten vorliegen (Gambardella, Taillard und Agazzi 1999, S. 12). Für alle Benchmarks beträgt die Anzahl von Knoten $n=100$.

Im Folgenden werden die Problemklassen als Vergleichskriterium genutzt, indem Mittelwerte aus den Benchmarks der entsprechenden Problemklasse gebildet werden, um einen übersichtlichen Vergleich zu gewährleisten.

Der im Folgenden dargestellte Vergleich beruht auf den Ergebnissen von

- **MACS**: (Gambardella, Taillard und Agazzi 1999, S. 12)
  - Beim Multiple ACS wird für jede Zielfunktion je eine Ameisenkolonie erzeugt, um die Zielfunktionen parallel optimieren zu können.
- **S-PSO**: (Gong, et al. 2012, S. 261)
  - Bei der set-based PSO wird die Position eines Partikels durch eine Teilmenge des Suchraums und die Geschwindigkeit durch Wahrscheinlichkeiten definiert.
- **RHGA**: (Berger, Barkaoui und Bräysy 2003, S. 17)
  - Beim Route-directed hybrid GA werden die genetischen Operatoren nicht auf kodierte Lösungen (Chromosome), sondern auf eine Population von Lösungen angewendet.
- **HSFLA**:(Luo, Li, et al. 2015, S. 281)
  - Der hybride SFLA beinhaltet neben dem herkömmlichen SFLA zudem eine Klon-Selektion[8], um die Qualität der Frösche zu verbessern.
- **Meta-HS**: (Yassen, et al. 2013, S. 637)
  - Beim Meta-HS Algorithmus werden durch einen sogenannten "Optimizer" die besten Werte für die HS-Parameter und für die lokale Suche bestimmt, welche anschließend von dem "Solver" zur Lösung des Problems verwendet werden.

Die Anwendung des HS Algorithmus beschränkt sich jedoch lediglich auf die Optimierung der Strecke und berechnet nicht die minimale Anzahl von benötigten Fahrzeugen. Zudem liegen seine durchschnittlichen Ergebnisse nur für einzelne Benchmarks vor, sodass er bei der Betrachtung der Problemklassen als Ganzes zunächst nicht berücksichtigt wurde (siehe **Tabelle 4-9**). Erst im Vergleich einzelner Benchmarks am Ende dieses Abschnittes, wird der HS Algorithmus ebenfalls eingebracht.

[8] Informationen zur Klon-Selektion sind zu finden in (Brunet 1959)

Tabelle 4-9: Gegenüberstellung der durchschnittlichen Ergebnisse für das VRPTW (Eigene Darstellung)

	MACS		S-PSO		RHGA		HSFLA	
Klasse	Fahrzeuge	Distanz	Fahrzeuge	Distanz	Fahrzeuge	Distanz	Fahrzeuge	Distanz
R1	12,00	1217,73	13,01	1263,25	12,17	1251,40	**11,92**	**1210,75**
R2	2,73	967,75	3,10	1073,72	2,73	1056,59	**2,70**	**951,51**
C1	**10,00**	**828,38**	**10,00**	856,44	**10,00**	828,50	**10,00**	**828,38**
C2	**3,00**	**589,86**	3,03	612,93	**3,00**	590,06	**3,00**	**589,86**
RC1	11,63	**1382,42**	12,66	1400,97	11,88	1414,86	**11,50**	1384,62
RC2	**3,25**	1129,19	3,59	1228,96	**3,25**	1258,15	3,30	**1119,63**

Aus dieser Gegenüberstellung geht hervor, dass der HSFLA insgesamt die besten Ergebnisse erzielt. Vor allem im Hinblick auf das Primärziel, der Minimierung der Fahrzeuganzahl, sind die Vorteile des HSFLA gegenüber den anderen Algorithmen zu erkennen. Auffällig ist zudem, dass bei einer gruppierten Kundenverteilung kaum Unterschiede bei der Lösungsgüte der Fahrzeugminimierung vorhanden sind, sodass bei der Problemklasse C die Streckenminimierung ausschlaggebend für die Wahl eines Algorithmus ist. In diesem Fall ist zwischen dem MACS und dem HSFLA zu wählen. Bei zufälliger bzw. kombinierter Kundenverteilung sollte auch der vorgeschlagene HSFLA gewählt werden.

Um die Ergebnisse besser einordnen zu können, ist in **Abbildung 4-6** und **Abbildung 4-7** die prozentuale Abweichung der durchschnittlichen Ergebnisse zum bekannten Optimum beider Zielfunktionen dargestellt.

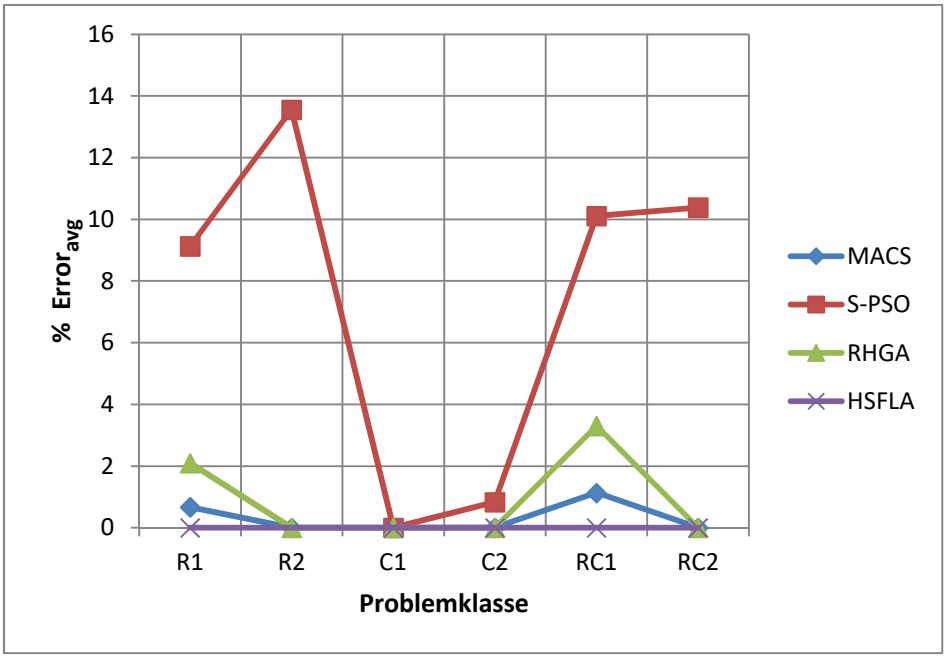


Abbildung 4-6: Prozentuale Abweichung zum Optimum hinsichtlich der Anzahl an Fahrzeugen (Eigene Darstellung)

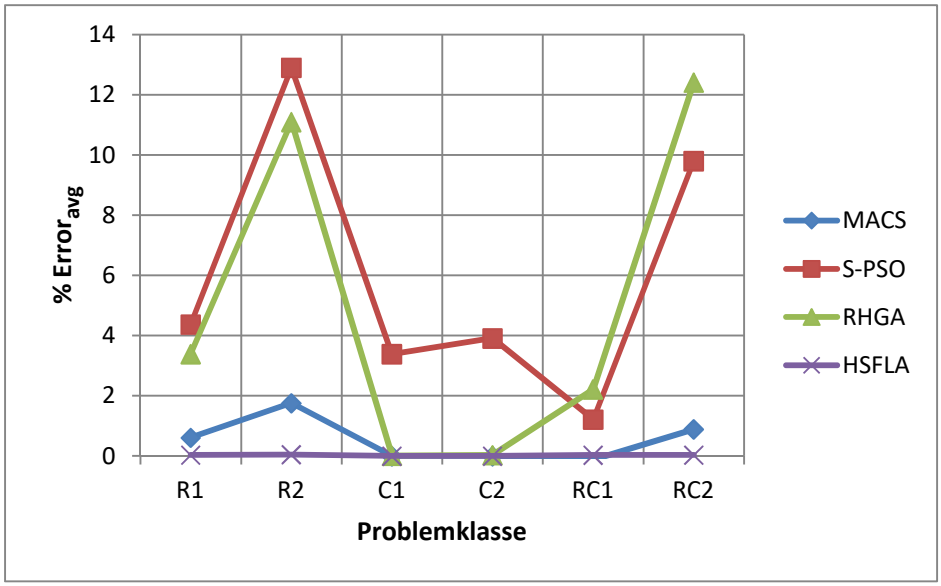


Abbildung 4-7: Prozentuale Abweichung zum Optimum hinsichtlich der Tourlänge (Eigene Darstellung)

Es ist zu erkennen, dass die S-PSO sowohl in Bezug auf die optimale Fahrzeuganzahl als auch auf die Tourlänge die höchsten Abweichungen aufweist. Dies ist unter anderem damit zu begründen, dass die vorliegende S-PSO der erste Algorithmus seiner Art war, welcher auf alle Benchmarks von Solomon angewendet wurde und somit noch recht unerforscht auf diesem Gebiet ist (Gong, et al. 2012, S. 264). Die anderen Algorithmen wurden diesbezüglich bereits näher untersucht und optimiert, was sich in ihrer Lösungsgüte widerspiegelt. Sie weisen nur geringe Abweichungen vom Optimum der primären Zielfunktion auf, weshalb die zweite Zielfunktion zur Entscheidungsfindung herangezogen wird. In diesem Fall erreicht der HSFLA in jeder Instanz die geringste Abweichung und ist daher zur Lösung für dieses Problem zu empfehlen. Doch auch das MACS sollte nicht gänzlich ausgeschlossen werden, da auch hier die Abweichungen minimal sind.

Im Folgenden werden einzelne Benchmarks der Problemklassen untersucht und der Meta-HS Algorithmus in den Vergleich mit einbezogen. Da keine Resultate in Bezug auf einzelne Benchmarks für das MACS und den RHGA vorliegen, sind diese beiden Verfahren kein Bestandteil dieses Vergleiches. Zudem fokussiert sich die Anwendung des HS Algorithmus ausschließlich auf die Minimierung der Transportwege und lässt die Optimierung der Fahrzeuganzahl außer Acht. In **Tabelle 4-10** sind die durchschnittlichen Ergebnisse des Meta-HS Algorithmus, HSFLA und der S-PSO für $n=100$ Knoten gegenübergestellt. Dabei wird mindestens eine Benchmark aus jeder Problemklasse untersucht. Die beiden letzten Ziffern in der Spalte "Instanz" kennzeichnen hierbei eine spezifische Benchmark der jeweiligen Problemklasse. So steht beispielsweise RC1-01 für die erste Benchmark der Problemklasse RC1.

Tabelle 4-10: Gegenüberstellung der durchschnittlichen Ergebnisse in Bezug auf die Streckenoptimierung beim VRPTW (Eigene Darstellung)

Instanz	Optimum	S-PSO	HSFLA	Meta-HS
R1-01	1645,79	1657,89	1650,98	**1644,01**
R1-03	1292,68	1268,95	1292,68	**1255,23**
R2-01	1252,37	1298,28	1252,88	**1232,95**
C1-02	828,94	850,84	**828,94**	831,67
C1-09	828,94	829,58	**828,94**	847,45
C2-06	588,49	592,95	**588,49**	632,87
C2-08	588,32	591,59	**588,32**	638,68
RC1-01	1696,94	1668,88	1697,23	**1663,50**
RC2-01	1406,91	1472,71	1407,22	**1385,50**

Aus dieser Gegenüberstellung geht hervor, dass die Stärke des Meta-HS Algorithmus auf VRPTW mit langen Strecken liegt. Bei einer Gesamtdistanz von über 1000 km erzielt dieser die besten Ergebnisse in diesem Vergleich. Dagegen werden bei Problemen mit einer Strecke von unter 1000 km die kürzesten Touren durch die Anwendung des HSFLA gefunden. Für eine bessere Einordnung ist eine illustrierte Darstellung der prozentualen Abweichung zum Optimum in **Abbildung 4-8** gegeben.

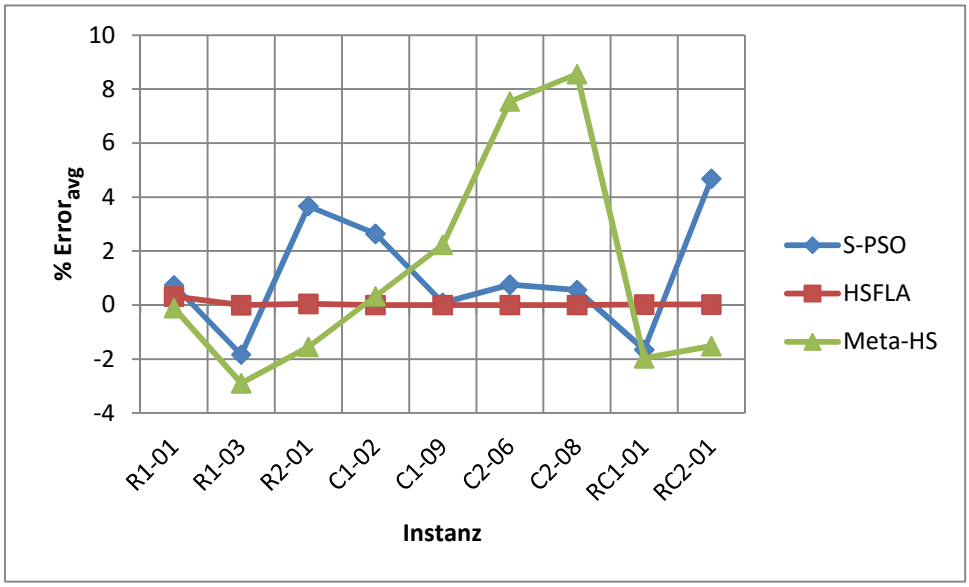


Abbildung 4-8: Gegenüberstellung der prozentualen Abweichung vom Optimum beim VRPTW (Eigene Darstellung)

Aus dieser Abbildung lassen sich zwei Schlussfolgerungen ziehen. Auf der einen Seite bestätigt sich der HSFLA als robuste Lösungsmöglichkeit für das VRPTW, da er über alle Instanzen nur geringe Abweichungen aufweist. Auf der anderen Seite können die S-PSO und der Meta-HS Algorithmus vereinzelt mit dem HSFLA in Bezug auf die Lösungsgüte mithalten, erweisen sich jedoch als zu inkonstant bei sich verändernder Problemstellung. Zwar können sie in den Instanzen R1-01 und RC1-01 kürzere Touren finden als in der Literatur (Solomon 2005) angegeben, jedoch überschreitet hierbei die Fahrzeuganzahl das beste bekannte

Minimum. Da, wie eingangs erwähnt, die Minimierung der Fahrzeuge das Primärziel ist, sind die Lösungen insgesamt dennoch schlechter als die Ergebnisse des HSFLA.

Multi Depot Vehicle Routing Problem

Um den Vergleich hinsichtlich der Vehicle Routing Probleme abzuschließen werden die effektivsten Algorithmen (ACS, GA und SFLA) in Bezug auf ein letztes VRP gegenübergestellt.

Bisher wurde davon ausgegangen, dass die Belieferung der Knoten von einem einzigen, zentralen Depot erfolgt. Beim MDVRP ist die Belieferung nun von mehreren unabhängigen Depots möglich. Das Ziel ist es, die minimale Distanz aller Fahrzeuge zu berechnen, indem die Depots den richtigen Kunden zugeordnet und optimale Touren berechnet werden. Die Eigenschaften der fünf bekanntesten Benchmarks dieser Problemstellung sind in **Tabelle 4-11** zusammengefasst und stammen aus der Onlinebibliothek der University of Málaga von (Díaz 2006).

Tabelle 4-11: Eigenschaften der betrachteten Benchmarks, in Anlehnung an (Surekha und Sumathi 2011, S. 124)

	Instanz				
Parameter	**P01**	**P02**	**P03**	**P04**	**P06**
Anzahl Kunden	50	50	75	100	100
Anzahl Depots	4	4	5	2	3
Anzahl Fahrzeuge	32	20	35	24	30
Fahrzeuge pro Depot	8	5	7	12	10
Fahrzeugkapazität	80	100	140	100	100

Die verwendeten Ergebnisse für diesen Vergleich stammen aus den Ausarbeitungen von

- PIACO: (Yu, Yang und Xie 2011)[9]
  - Parallel Improved Ant Colony optimization
- GA: (Surekha und Sumathi 2011, S. 130)
  - Herkömmlicher GA (vgl. Abschnitt 3.3.3)
- MPMSFLA: (Luo und Chen 2014, S. 94)
  - Der Multi-phase modified SFLA verbindet den herkömmlichen SFLA mit der "K-means-Methode" zur Clusteroptimierung

Anzumerken bei dem Vergleich in **Tabelle 4-12** und **Abbildung 4-9** ist, dass nicht die durchschnittlichen Ergebnisse aus mehreren Durchläufen, sondern das beste erreichte Ergebnis als Vergleichsgrundlage gilt.

[9] Die eigentliche Ausarbeit ist nicht für die Öffentlichkeit zugänglich, weshalb die Ergebnisse aus einer Zusammenfassung von (Karakatič und Podgorelec 2015, S. 530) stammen und keine genaueren Erläuterungen zur Vorgehensweise vorliegen

Tabelle 4-12: Gegenüberstellung der besten erzielten Ergebnisse beim MDVRP (eigene Darstellung)

Instanz	Optimum	PIACO	GA	MPMSFLA
P01	576,87	**576,87**	622,18	**576,87**
P02	473,53	**473,53**	480,04	**473,53**
P03	641,15	**641,19**	706,88	**641,19**
P04	1001,04	**1001,04**	1024,78	**1001,04**
P06	876,50	876,70	908,88	**876,50**

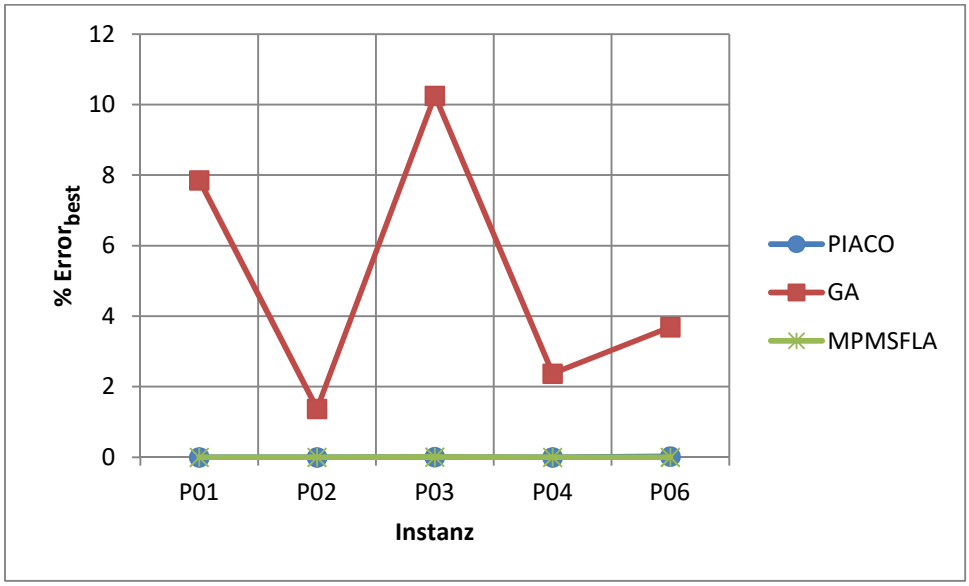


Abbildung 4-9: Gegenüberstellung der Abweichung der besten erzielten Ergebnissen vom Optimum (Eigene Darstellung)

Tabelle 4-12 und **Abbildung 4-9** zeigen, dass sowohl der MPMSFLA und als auch die PIACO mit Ausnahme von P06 in der Lage sind, die optimale Tour zu berechnen. Ihre gute Lösungsgüte ist dadurch zu erklären, dass sie für das MDVRP angepasst bzw. verbessert sind. Dahingegen werden beim GA keine Verbesserungen vorgenommen, weshalb nur der Basisalgorithmus verwendet wird. Dies lässt viel Spielraum für Erweiterungen und erklärt die schlechteren Lösungen im Vergleich zum PIACO respektive MPMSFLA. Allerdings spiegelt dieser Vergleich auch die Erkenntnisse aus den vorangegangenen Vergleichen in Bezug auf das VRP wieder, aus denen hervorging, dass die Variationen des Ameisenalgorithmus und des SFLA insgesamt die besten Ergebnisse lieferten und der GA etwas schlechtere Lösungen erzielt.

4.3.3 Warehouse Location Problem als Benchmark

Nachdem der Fokus der Vergleiche in den vorangegangenen Abschnitten auf den Tourenplanungen lag, wird nun die Lösungsgüte der Algorithmen in Bezug auf Standortprobleme untersucht. Diese bilden die zweite große Problemstellung im Güterfluss logistischer Netzwerke, da die Wahl eines optimalen Standortes der zweite wichtige Aspekt für eine kostenminimale Belieferung der Akteure innerhalb eines logistischen Netzwerkes ist (vgl. Abschnitt 2.3.3). Weil in Abschnitt 4.4.4 eines der komplexesten bekannten

Standortprobleme als Vergleichsgrundlage dient, wird in diesen Teil der Arbeit das weniger komplexe unkapazitierte Warehouse Location Problem (UWLP) betrachtet. Beim UWLP wird von einer unendlichen Kapazität der Fabriken ausgegangen und es müssen nur wenige Nebenbedingungen beachtet werden. Gleichzeitig ist das UWLP das in der Literatur am meisten untersuche Warehouse Location Problem (Maric 2010, S. 184), weshalb insgesamt gute Ergebnisse zu erwarten sind. Allerdings existieren keine Daten für den SFLA in Bezug auf Standortplanungen, weswegen dieser unberücksichtigt bleibt. Die genutzten Benchmarks entstammen aus der Onlinebibliothek "OR-Library" und sind zu finden auf (Beasly 2018).

Unkapazitiertes Warehouse Location Problem

Um die Ergebnisse der Algorithmen für möglichst viele unterschiedliche Probleme erfassen zu können, wurden sie auf Benchmarks verschiedener Problemgrößen angewandt. **Tabelle 4-13** liefert einen Überblick über die Charakteristika der einzelnen Benchmarks. Dabei wird ihre Größe durch die Anzahl der potentiellen Standorte m und die Anzahl der Kunden n festgelegt.

Tabelle 4-13: Überblick über die Benchmarks des UWLP (Guner und Sevkli 2008, S. 5)

Instanz	Größe (m x n)	Optimum
Cap71	16 x 50	932.615,75
Cap72	16 x 50	977.799,40
Cap73	16 x 50	1.010.641,45
Cap74	16 x 50	1.034.976,98
Cap101	25 x 50	796.648,44
Cap102	25 x 50	854.704,20
Cap103	25 x 50	893.782,11
Cap104	25 x 50	928.941,75
Cap 131	50 x 50	793.439,56
Cap132	50 x 50	851.495,33
Cap 133	50 x 50	893.076,71
Cap 134	50 x 50	928.941,75
CapA	100 x 1000	17.156.454,48
CapB	100 x 1000	12.979.071,58
CapC	100 x 1000	11.505.594,33

Es ist zu erkennen, dass 12 der 15 Benchmarks kleine Problemstellungen mit nur 50 zu beliefernden Akteuren sind, wohingegen die Benchmarks CapA bis CapC sehr große Probleme mit 1000 Kunden sind. Welche Algorithmen sich für Probleme dieser Größe eignen und ob es Unterschiede in der Genauigkeit der Lösungen bei wachsender Problemstellung gibt, wird im Folgenden untersucht.

Dabei beruht der Vergleich auf den Ergebnissen von

- **ACS**: (Kole, Chakrabarti und Bhattacharyya 2014, S. 60)
    - Herkömmliches Ant Colony System (vgl. Abschnitt 3.1.3)

- **DPSO$_{LS}$**: (Guner und Sevkli 2008, S. 6-7)
  - Dies ist eine Kombination aus einer diskreten PSO, bei der die Partikel mithilfe einer binären Kodierung nur aus einem Vektor zur Entscheidung der Standorteröffnung bestehen und der lokalen Suche.
- **GA**: (Jaramillo, Bhadury und Batta 2002)[10]
  - Herkömmlicher Genetischer Algorithmus (vgl. 3.3.3)
- **OBCHS**: (Heidari, Kazemizade und Abbaspour 2015, S. 309-310)
  - Beim "opposition-based Chaotic HS Algorithmus" wird während der Initialisierungsphase für jede potentielle Lösung $Y = (y_1^1, \dots, y_1^n)$ ein Gegenwert $\tilde{Y} = (\breve{y}_1^1, \dots, \breve{y}_1^n)$ erzeugt und überprüft, ob dadurch eine bessere Fitness erzeugt werden kann. Fall ja wird Y durch $\tilde{Y}$ ersetzt, ansonsten wird Y beibehalten.

Jeder dieser Algorithmen wurde während der Fallstudien mehrmals zur Lösung der einzelnen Benchmarks eingesetzt und Mittelwerte aus den Ergebnissen gebildet. Das ACS wurde jedoch nicht für die großen Probleminstanzen CapA bis CapC eingesetzt, weshalb hierfür keine Ergebnisse vorliegen. Warum der Einsatz des ACS für diese Probleminstanzen nicht sinnvoll ist, zeigt **Abbildung 4-10**. Dort werden die Abweichungen der gemittelten Ergebnisse zum Optimum dargestellt.

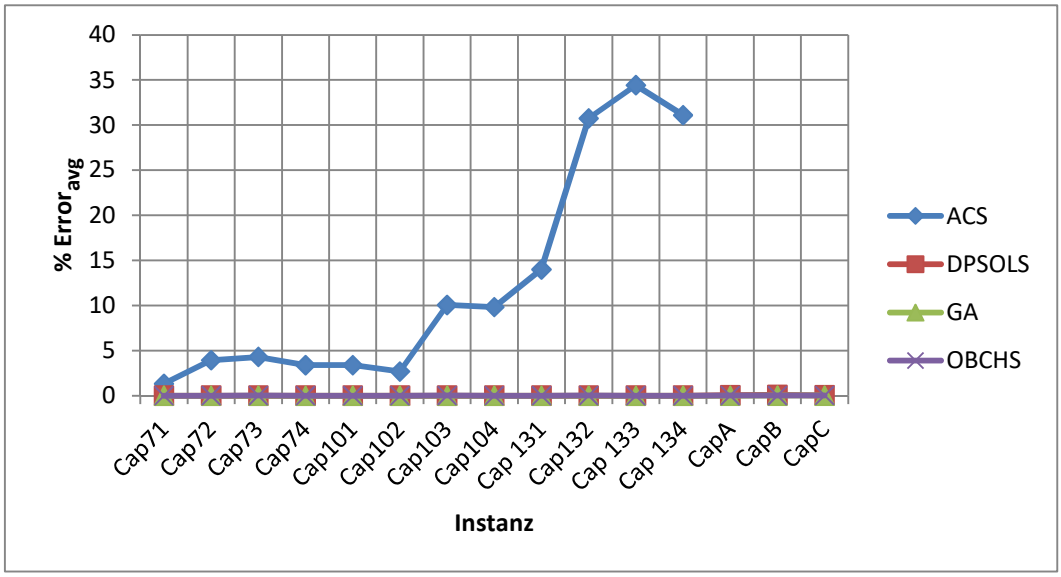


Abbildung 4-10: Gegenüberstellung der prozentualen Abweichung zum Optimum beim UWLP (Eigene Darstellung)

Aus **Abbildung 4-10** ist deutlich zu entnehmen, dass sich das ACS nicht für UWLP eignet. Dies ist damit zu erklären, dass die beschriebene Fallstudie die erste Anwendung des ACS für das UWLP war und somit noch keine Modifizierungen oder Verbesserungen entwickelt wurden. Zwar liegen die Ergebnisse für kleinere Problemstellungen noch in einem akzeptablen Bereich, doch wird dieser mit wachsender Problemgröße zunehmend ungenauer und weist

[10] Der GA für das UWLP wurde von den genannten Autoren entwickelt. Die Ergebnisse für die in Tabelle 4-14 erwähnten Benchmarks stammen allerdings von (Guner und Sevkli 2008, S. 7)

schon für UWLP der Größe 50 x 50 Abweichungen von bis zu 34% auf. Dahingegen erweisen sich die weiteren Algorithmen als robuste Lösungsmöglichkeiten und weisen auch für die größten Probleme nur minimale Abweichungen zum Optimum auf. Für einen genaueren Vergleich wird das ACS nicht weiter betrachtet, sodass die Ergebnisse der übrigen Algorithmen durch eine feinere Beschriftung der Y-Achse in **Abbildung 4-11** detaillierter verglichen werden können.

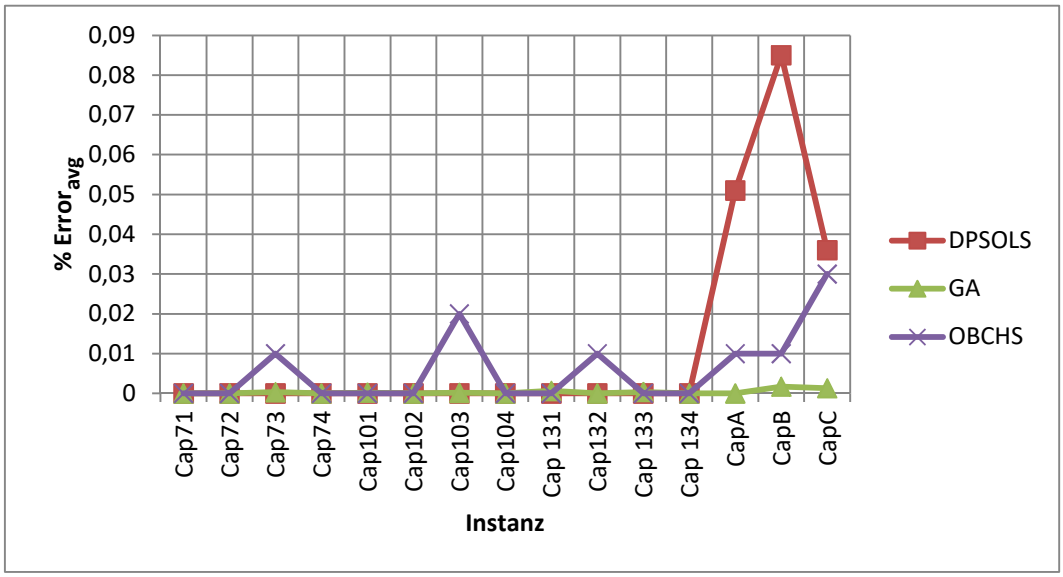


Abbildung 4-11: Gegenüberstellung der prozentualen Abweichung zum Optimum beim UWLP (Ohne ACS) (Eigene Darstellung)

Aus dieser Gegenüberstellung erweist sich der GA als bestes Verfahren zur Lösung des UWLP. Selbst bei den großen Problemstellungen mit 1000 zu beliefernden Kunden liegt die Abweichung bei maximal 0,00172%. Bei der $DPSO_{LS}$ und dem OBCHS Algorithmus werden die Ungenauigkeiten mit zunehmender Problemstellung größer. Jedoch liegt die maximale Abweichung bei unter 0,09%, was alle drei genannten Algorithmen zu sehr guten Lösungsverfahren in Bezug auf dieses Problem macht. Damit bestätigt sich die Erwartung, dass auch große UWLP effizient und annähernd optimal lösbar sind.

4.3.4 Quadratic Assignment Problem als Benchmark

Um den Vergleich der Algorithmen hinsichtlich der Problemstellungen in logistischen Netzwerken abzuschließen, wird das QAP als Vergleichsgrundlage herangezogen. Da hierbei der Güterfluss zwischen den Betrieben untereinander relevant ist und der Standort für alle Betriebe wegoptimal gewählt werden muss, besteht keine Möglichkeit sich an der geographischen Lage der Kunden zu orientieren. Allen involvierten Betrieben muss ein Standort zugeordnet werden, sodass sich $n!$ mögliche Kombinationen ergeben. Dies macht das Problem zu einem der komplexesten Optimierungsprobleme (Amudha und Shivakumar 2012, S. 3), (R. E. Burkard 2013, S. 243-244). Der folgende Vergleich beschränkt sich auf Benchmarks mit $n \geq 20$ Betrieben, da kleinere Probleme leicht mit exakten Verfahren gelöst werden können.

Die verwendeten Benchmarks sind in der Onlinebibliothek von (Burkard, Cela, et al. 2018) zu finden. Allgemein lassen sich diese Benchmarks in vier Kategorien einteilen. Die erste Kategorie beinhaltet zufällig erstelle Benchmarks mit symmetrischen Güterflussmatrizen. D.h. Betrieb i erhält von Betrieb j die gleiche Menge an Gütern wie j von i. In der zweiten Kategorie sind die Distanzen als City-Block-Distanz angegeben (vgl. Abschnitt 3.1.3). Die Benchmarks der Kategorie drei sind realitätsnahe Problemstellungen mit asymmetrischer Güterflussmatrix. Die letzte Kategorie umfasst reale Problemstellungen mit unterschiedlichen Merkmalen.

Der durchgeführte Vergleich beruht auf den Ergebnissen von

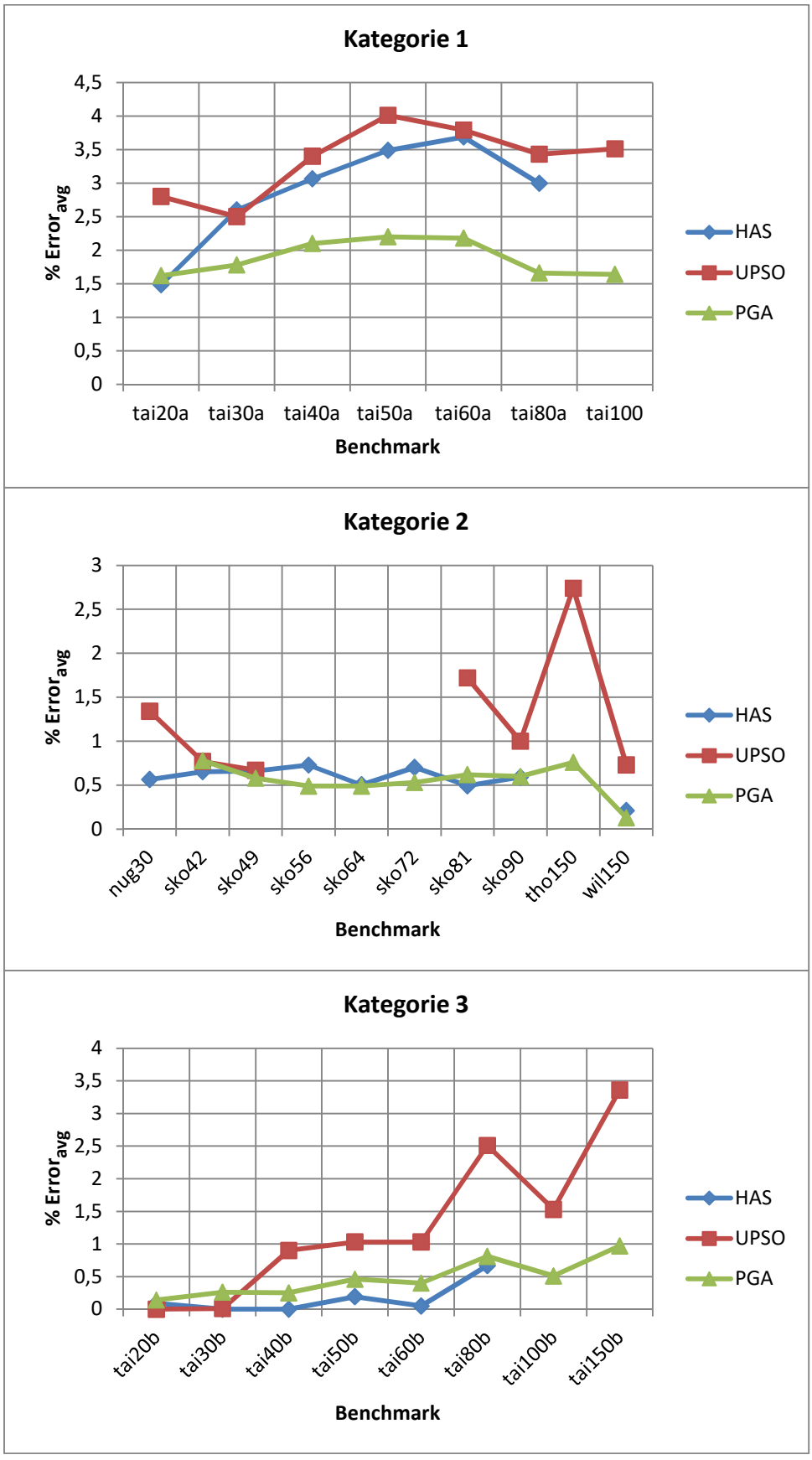
- **HAS**: (Gambardella, Taillard und Dorgio 1999, S. 173-174)
  - Beim hybriden Ant System existiert nur ein globales Pheromonupdate, was zu einer schnelleren Erreichung des Optimums führt. Um dabei eine vorzeitige Konvergenz zu verhindern, werden nach einer gewissen Zeit alle Pheromonkonzentrationen auf den Kanten gelöscht. Desweiteren wird die Pheromonspur nicht zur Erforschung neuer Wege genutzt, sondern die existierenden Lösungen ähnlich wie bei einer Nachbarschaftssuche verändert.
- **UPSO**: (Hafiz und Abdennour 2016, S. 427-428)[11]
  - Die diskrete "*unified PSO*" vereint die l_{best} und g_{best}-Nachbarschafts-Methode (vgl. Kapitel 3.2.2) durch einen Vereinigungsfaktor u.
- **PGA** : (Lim, Yuan und Omatu 2002, S. 55-62)
  - Der "*partial local search GA*" ist eine Kombination des herkömmlichen GA und der lokalen Suche, bei der Gene miteinander vertauscht werden.
- **HICGS**: (Amudha und Shivakumar 2012, S. 6)
  - Beim "*hybrid Harmony Improvised Consultant Guided Search*" werden die Lösungen der *Consultant Guided Search*-Methode[12] durch den HS Algorithmus erzeugt.

Für den HICGS Algorithmus liegen allerdings nur die besten erreichten Ergebnisse für kleine bis mittlere Problemstellungen mit $n \leq 50$ Betrieben vor, sodass er nicht Teil des folgenden Vergleiches ist. Eine komprimierte Zusammenfassung der Erkenntnisse des HICGS Algorithmus für das QAP wird am Ende des Kapitel wiedergegeben.

Um die Effektivität der Algorithmen für unterschiedliche Problemstellungen zu veranschaulichen, werden die vier eingangs erwähnten Kategorien getrennt voneinander betrachtet. Aus jeder Kategorie werden Benchmarks ausgewählt und ihr Optimum mit den durchschnittlichen Ergebnissen der Algorithmen verglichen. Daraus ergeben sich vier unabhängige Grafiken, welche die Abweichung der durchschnittlichen Ergebnisse vom Optimum gegenüberstellen (siehe **Abbildung 4-12**). Die mitunter auftretenden

[11] Der Algorithmus wurde ursprünglich von (Parsopoulos und Vrahatis 2004) entwickelt
[12] Nähere Informationen zur Consultant Guided Search-Methode sind zu finden in (Iordache 2010, S. 225-227)

Unterbrechungen der Kurven resultieren aus der Tatsache, dass nicht für alle Benchmarks die entsprechenden Ergebnisse vorliegen.



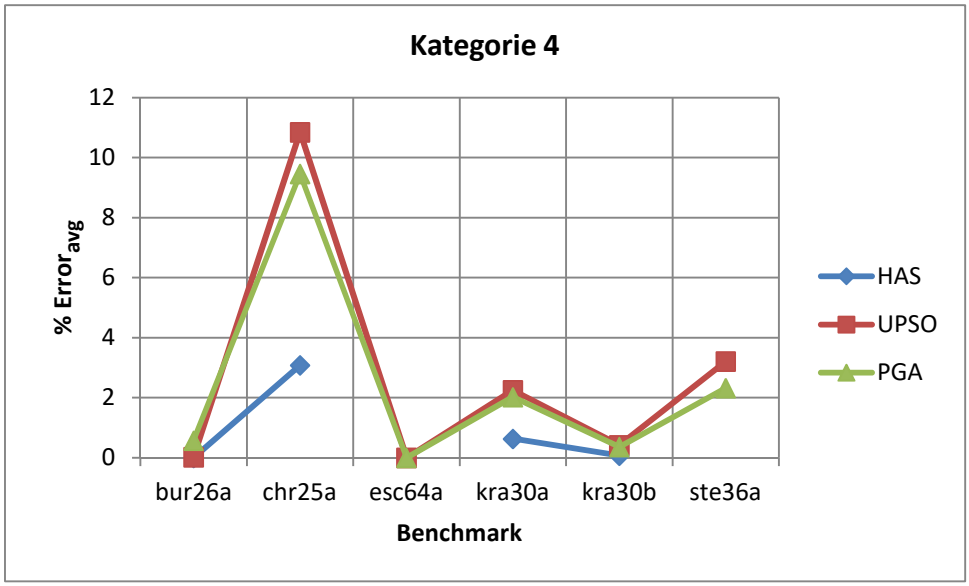


Abbildung 4-12: Prozentuale Abweichung der durchschnittlichen Ergebnisse zum Optimum für vier unabhängige Problemkategorien (Eigene Darstellung)

Bei Betrachtung der einzelnen Graphen lässt sich feststellen, dass der PGA bei Problemen mit symmetrischem Güterfluss insgesamt die besten Ergebnisse liefert. Mit einer Ausnahme bei tai20a liefert er deutlich bessere Ergebnisse als die übrigen zwei Algorithmen. Zudem erweist er sich als sehr stabiles Verfahren, welcher auch bei wachsender Problemgröße nur geringfügig ungenauer wird. Für die Probleme der Kategorie 2 erweisen sich das HAS und der PGA als in etwa gleich effektive Lösungsverfahren mit stabilen Ergebnissen über jede Instanz. In den Kategorie 3 mit asymmetrischem Güterfluss erzielen alle Algorithmen für kleine Probleme gute Lösungen, doch nur das HAS bleibt bei steigender Knotenanzahl konstant und liefert Ergebnisse nahe des Optimums. Insbesondere die UPSO wird zunehmend ungenauer je mehr Akteure zu beliefern sind. In dem letzten Graph sind exemplarisch die Ergebnisse für praktische QAP illustriert. Dabei spielt die Anzahl der Knoten in Bezug auf die Komplexität nur eine untergeordnete Rolle. Vielmehr ist hierbei entscheidend, welches Problem die jeweiligen Benchmarks beschreiben. So kann das QAP beispielsweise auch für den Entwurf von Schreibmaschinentastaturen, in der Graphentheorie oder auch zur Layoutplanung angewendet werden. Dies erklärt das sprunghafte Verhalten der Abweichung im Graphen der Kategorie 4. Dennoch ist auch hier der Trend zu erkennen, dass das HAS für die verschiedenen Problemarten die besten Ergebnisse erzielt. Durch eine Zusammenfassung der Erkenntnisse aus allen vier Kategorien, lässt sich erschließen, dass der PGA für symmetrische QAP im Vorteil ist, während vor allem bei realitätsnahen bzw. praktischen Problemen das HAS angewendet werden sollte. Die UPSO liefert in allen Kategorien die schlechtesten Ergebnisse, weshalb diese nur bedingt für Probleme dieser Art geeignet ist.

Für den HICGS liegen nur die besten erzielten Ergebnisse für Benchmarks mit $n \leq 50$ Knoten vor. Es wurden jeweils sechs Benchmarks aus jeder Kategorie ausgewählt und Mittelwerte gebildet, sodass in **Abbildung 4-13** die Abweichungen in Bezug auf das Optimum der vier Kategorien dargestellt ist.

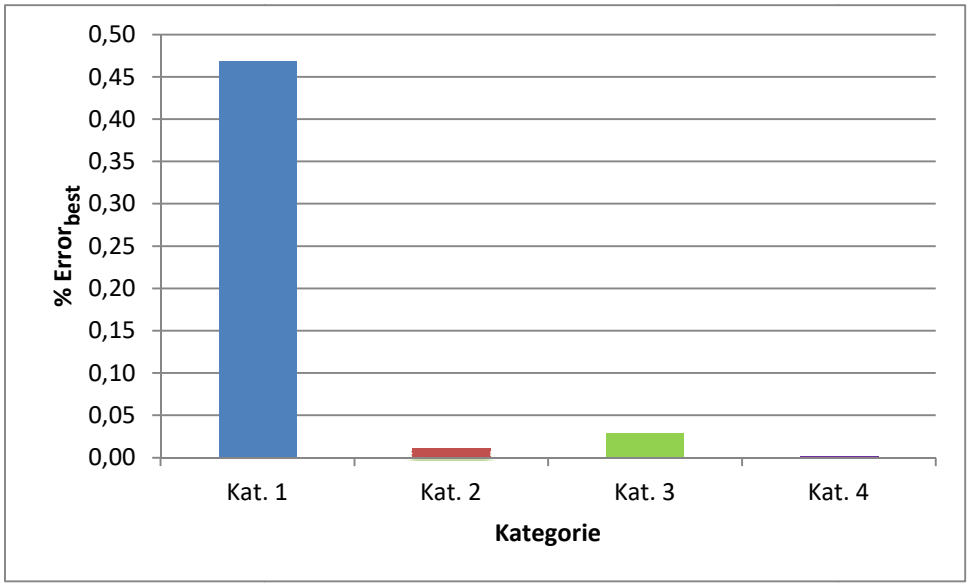


Abbildung 4-13: Abweichung der besten erzielten Ergebnisse des HICGS zum Optimum (Eigene Darstellung)

Aus **Abbildung 4-13** ist zu entnehmen, dass der HICGS Algorithmus in der Lage ist, in jeder Kategorie sehr nah an das globale Optimum zu gelangen. In Kategorie 4 erreicht der Algorithmus sogar in allen Instanzen das Optimum. Allerdings muss einschränkend gesagt werden, dass keinerlei Informationen über die Anzahl der Durchläufe oder die benötigte Laufzeit vorliegen. Desweiteren ist das Verhalten des Algorithmus bei größeren Problemen mit $n>50$ Knoten unbekannt. Nichtsdestotrotz liefert dieser Algorithmus vielversprechende Lösungen, was ihn zu einem geeignetem Kandidaten für weitere Untersuchungen macht.

Aus den untersuchten Benchmarks folgt, dass, mit Ausnahme des SFLA für Standortprobleme, alle beschriebenen Algorithmen zulässige Lösungen für die betrachteten Problemstellungen berechnen können. Für eine strukturierte Übersicht über die Lösungsgüte der Algorithmen in Bezug auf unterschiedliche Probleme wird im Nächsten Kapitel eine Kategorisierung durchgeführt.

5 Kategorisierung der Algorithmen

Nachdem alle Algorithmen hinsichtlich der Anwendungsgebiete in logistischen Netzwerken untersucht und verglichen wurden, erfolgt in diesem Kapitel eine Kategorisierung. Dazu werden unter Berücksichtigung der gewonnen Erkenntnisse des vorherigen Kapitels die Eigenschaften der Problemstellungen zusammengefasst und die Probleme anhand ihrer Charakteristiken gruppiert, sodass hervorgehoben kann, welche Algorithmen zur Lösung welcher Probleme eingesetzt werden können. Da Touren- und Standortplanungen zwei grundsätzlich verschiedene Optimierungsprobleme darstellen, erfolgt jeweils eine Kategorisierung für jede der beiden Problemstellungen, um zum Schluss eine differenzierte Übersicht über die Leistung der Algorithmen erarbeiten zu können. Die genaue Einteilung der Benchmarks kann im Anhang C nachgeschlagen werden.

5.1 Unterscheidungskriterien für die Kategorisierung

Um besser nachvollziehen zu können, nach welchen Merkmalen die Problemstellungen zusammengefasst werden, sind in diesem Abschnitt die Parameter nach denen die Kategorisierung erfolgt dargestellt.

Das erste entscheidende Merkmal, nach dem die Probleme zusammengefasst werden, ist die **Größe** der betrachteten Problemstellungen. Dabei bezieht sich die Größe auf die Anzahl der Knoten n. Für den weiteren Verlauf dieser Arbeit werden folgende Größen definiert:

- Klein: $n \leq 50$
- Mittel: $50 < n \leq 100$
- Groß: $n < 100$

Zudem werden die Probleme in Bezug auf ihre **Komplexität** eingeteilt. Da alle betrachteten Problemstellungen NP-schwer bzw. NP-vollständig sind, steht die hier definierte Komplexität nicht mit der Größe des Suchraums, sondern mit der Anzahl der Nebenbedingungen im Zusammenhang. Sowohl für die Touren- als auch für die Standortplanung wurden in den vorangegangenen Kapiteln "einfache" und "komplexe" Problemstellungen beschrieben. So fällt das TSP aufgrund seiner trivialen Nebenbedingungen in die Kategorie "einfach", während das CVRP, VRPTW und MDVRP die komplexen Probleme für Tourenplanungen darstellen. Bei der Standortplanung wurden das UWLP und das QAP als Beispiele für einfache bzw. komplexe Probleme vorgestellt.

Als Letztes wird zwischen der **Anzahl der Zielfunktionen** der einzelnen Problemstellungen unterschieden. Dazu werden alle Benchmarks anhand der zu minimierenden Zielfunktionen gruppiert, um untersuchen zu können, welche Algorithmen mit Blick auf alle Zielfunktionen die besten Ergebnisse liefern.

5.2 Kategorisierung hinsichtlich Tourenplanungen

Als erstes werden die Herausforderungen bei Tourenplanungen in Bezug auf die genannten Charakteristiken zusammengefasst und die Güte der Algorithmen dargestellt. Dazu wurden alle Benchmarks der vorherigen Kapitel anhand ihrer Eigenschaften eingeteilt, damit als Resultat Grafiken erstellt werden können, in der die Algorithmen hinsichtlich der erwähnten Parameter bewertet werden.

5.2.1 Einfache Tourenplanungen

Im Folgenden sind jeweils einzelne Grafiken für die einfacheren und die komplexen Problemstellungen dargestellt. In **Abbildung 5-1** werden die Gesamtabweichungen der Algorithmen für einfache Tourenplanungen verschiedener Größe vorgestellt. Da die betrachteten Benchmarks nur eine Zielfunktion aufweisen, muss in dieser Hinsicht nicht unterschieden werden.

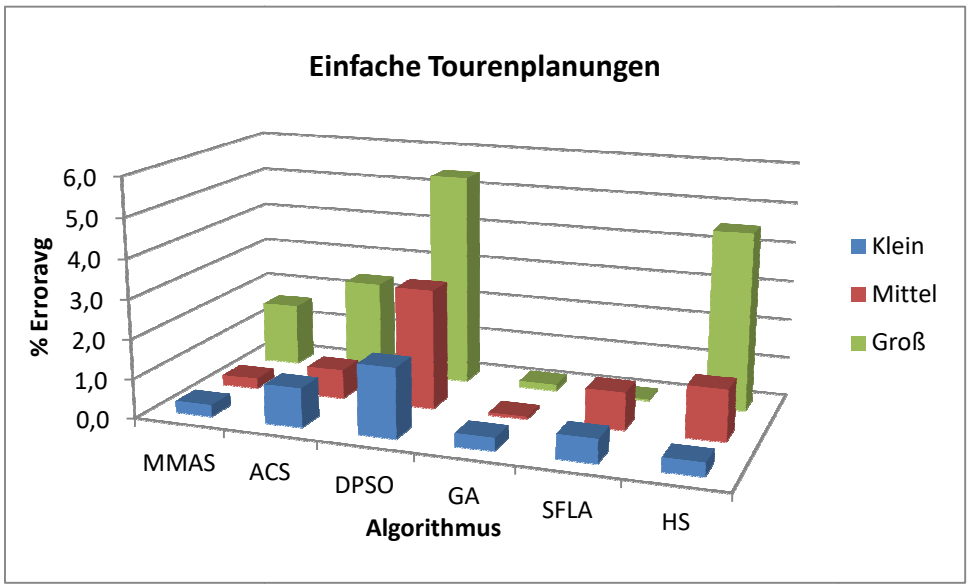


Abbildung 5-1: Durchschnittliche Abweichungen zum Optimum bei einfachen Tourenplanungen (Eigene Darstellung)

Zur Lösung dieser Probleme werden hauptsächlich die Algorithmen in ihrer Basisform verwendet. Eine Ausnahme bildet der SFLA, bei dem für große Problemstellungen eine Verbesserung vorgenommen wird, was die gute Lösungsgüte in diesem Punkt erklärt (vgl. Abschnitt 4.4.1). Dass bei der PSO eine Diskretisierung notwendig ist, wurde in den vorangegangenen Kapiteln ausführlich erläutert, weshalb dies in diesem Vergleich nicht als Verbesserung anzusehen ist.

Durch einen Vergleich der dargestellten Ergebnisse fällt auf, dass der GA die besten Lösungen erzielt und auch bei wachsender Knotenanzahl zuverlässig gute Ergebnisse liefert. Klammert man die Verbesserung des SFLA aus, so ist der Trend zu erkennen, dass die übrigen Algorithmen zunehmend ungenauer werden, je größer das Problem wird. Für kleinere Probleme liefern alle Algorithmen gute zulässige Lösungen, doch falls möglich sollte stets der GA zur Lösung dieses Problems angewandt werden.

5.2.2 Komplexe Tourenplanungen

In diesem Abschnitt werden Tourenplanungen betrachtet, bei denen mehr Restriktionen durch Nebenbedingungen berücksichtigt werden müssen. Dies führt zu einer komplexeren Aufgabenstellung, welche die Algorithmen lösen müssen. Um der gestiegenen Komplexität entgegenwirken zu können, wurden die Algorithmen weiterentwickelt und verbessert. Dieser Vergleich beinhaltet aus diesem Grund hauptsächlich modifizierte Versionen der Basisalgorithmen. Im Gegensatz zum vorherigen Vergleich, wird in **Abbildung 5-2** nun auch zwischen der Anzahl der Zielfunktionen bei großen Problemstellungen differenziert. Existiert für eine Problemstellung keine zweite Zielfunktion, so wird die einzige Zielfunktion als primäres Ziel gewertet. Da nicht für alle Algorithmen und Problemstellungen Ergebnisse vorliegen, sind mache Bereiche des Diagramms nicht ausgefüllt. Desweiteren wird, wie in Abschnitt 4.4.2 bereits erwähnt, das MMAS aufgrund seiner langen Laufzeit bei komplexen Problemen nicht weiter betrachtet und nur das ACS impliziert.

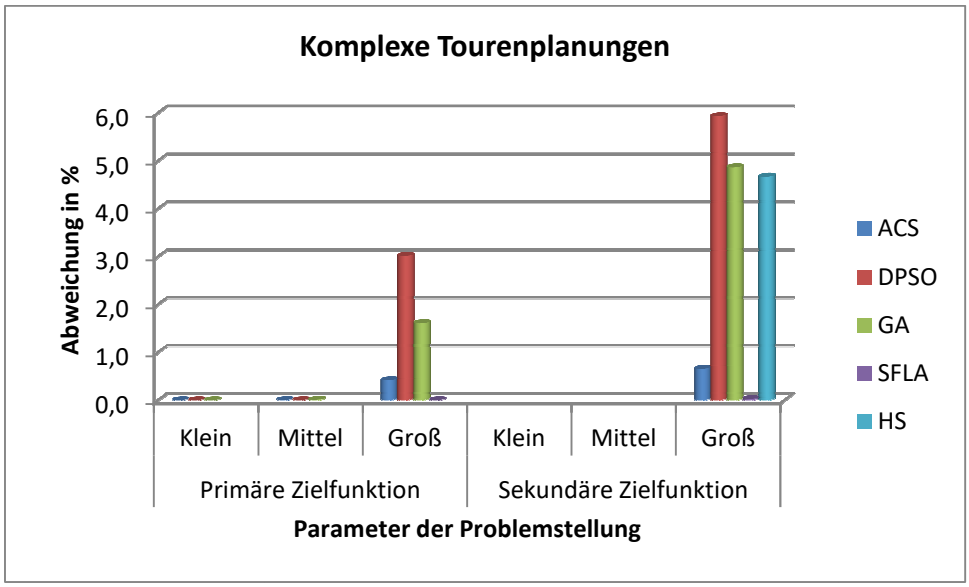


Abbildung 5-2: Durchschnittliche Abweichungen zum Optimum bei komplexen Tourenplanungen (Eigene Darstellung)

Dieser Vergleich verdeutlicht, dass für kleine und mittlere Probleme alle untersuchten Algorithmen sehr gute Lösungen erzielen. Für kleine Probleme finden alle Algorithmen sogar in jedem Durchgang das globale Optimum, während bei Problemen mittlerer Größe die Abweichung bei maximal 0,0099% liegt und damit so klein ist, dass sie kaum einen Einfluss auf die Lösungsgüte hat. Bei großen Problemen konnte zusätzlich noch zwischen der Anzahl an Zielfunktionen differenziert werden. Hieraus ist die Effizienz von modifizierten Algorithmen gut zu erkennen. Während im vorherigen Vergleich durch den GA die besten Ergebnisse erreicht werden konnten, finden nun die weiterentwickelten Versionen des SFLA die optimalsten Touren. Insbesondere bei Problemstellungen mit mehreren Zielfunktionen wird die gute Performance des SFLA deutlich. Während sich bei allen anderen Algorithmen die Werte der sekundären Zielfunktion im Gegensatz zur primären verschlechtern, liefert der SFLA konstant gute Ergebnisse unabhängig von der Größe oder der Anzahl an Zielfunktionen. Hervorzuheben ist hierbei auch der HS Algorithmus. Bei Anwendung dieses Algorithmus wird

die primäre Zielfunktion außer Acht gelassen und nur eine Streckenoptimierung vorgenommen. Trotzdem werden nur relativ schlechte Ergebnisse erzielt, weshalb er nicht für komplexe Tourenplanungen eingesetzt werden sollte.

Insgesamt bestätigt sich hierdurch der Eindruck, dass das ACS, der GA und der SFLA die besten Ergebnisse für Tourenplanungen erzielen und die DPSO sowie der HS Algorithmus weitere Verbesserungen benötigen, um für reale Probleme eingesetzt werden zu können.

5.3 Kategorisierung hinsichtlich Standortplanungen

Analog zu Kapitel 5.2 werden in diesem Abschnitt die Standortprobleme in einfache und komplexe Probleme eingeteilt und die erzielten Ergebnisse der Algorithmen in Bezug auf die Problemstellungen miteinander Verglichen.

5.3.1 Einfache Standortprobleme

Als einfaches Standortproblem wird das UWLP ausgewählt. Obwohl es eine der meist untersuchtesten Problemstellungen in logistischen Netzwerken ist, liegen keine bekannten Anwendungen des SFLA vor. Dies ist ein Indiz dafür, dass der SFLA nach dem heutigen Stand der Technik noch nicht für Standortprobleme geeignet ist und weitere Untersuchungen auf diesem Gebiet notwendig sind. Nichtsdestotrotz sind die Ergebnisse der übrigen Algorithmen in **Abbildung 5-3** gegenübergestellt. Dazu werden die Benchmarks nach bekanntem Muster ihren Eigenschaften entsprechend in Gruppen eingeteilt. Alle Probleme enthalten nur eine Zielfunktion, weshalb eine Unterteilung nach diesem Parameter nicht notwendig ist. Desweiteren liegen keine Ergebnisse für mittlere Problemstellungen vor und können somit nicht betrachtet werden.

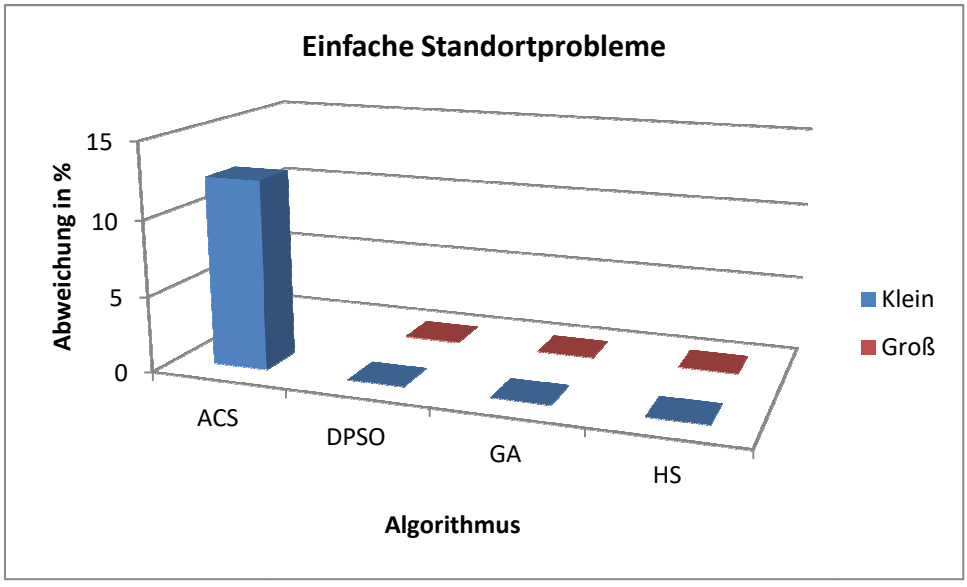


Abbildung 5-3: Durchschnittliche Abweichungen zum Optimum bei einfachen Standortproblemen (Eigene Darstellung)

Aus dieser Gegenüberstellung geht hervor, dass das ACS ohne Modifikationen nicht als Lösungsmöglichkeit für Standortprobleme geeignet ist. Schon für kleine Problemstellungen beträgt die durchschnittliche Abweichung zum Optimum über 10%, weshalb dieser Algorithmus nicht zur Lösung für große Standortprobleme angewendet wurde. Im Gegensatz dazu erzielen die übrigen Algorithmen unabhängig von der Problemgröße sehr gute Lösungen. Obwohl auch der GA in seiner Basisversion eingesetzt wurde, erreicht dieser bessere Lösungen als der modifizierte HS Algorithmus und die DPSO und erzielt somit erneut die besten Ergebnisse.

5.3.2 Komplexe Standortprobleme

Die letzte Vergleichsgruppe bilden die komplexen Standortprobleme. Auch hier findet der SFLA keine Anwendung und kann somit nicht in den Vergleich mit einbezogen werden. Zusätzlich kann beim HS Algorithmus nur seine grundsätzliche Tauglichkeit für dieses Problem bestätigt werden, ohne dass dieser intensiv für einzelne Benchmarks getestet wurde (vgl. Abschnitt 4.4.4). Somit beschränkt sich der Vergleich in **Abbildung 5-4** auf weiterentwickelte Versionen des ACS, GA und der DPSO.

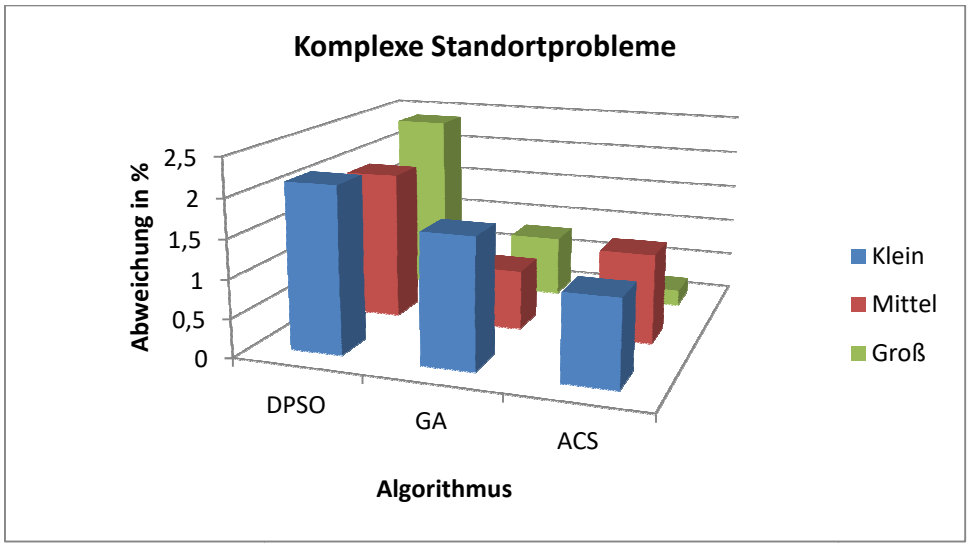


Abbildung 5-4: Durchschnittliche Abweichungen zum Optimum bei komplexen Standortproblemen (Eigene Darstellung)

Diese Abbildung unterstreicht erneut die gute Performance von modifizierten Algorithmen. Trotz der gestiegenen Komplexität des QAP erzielen mit einer maximalen Abweichung von 2,5% alle Algorithmen gute Ergebnisse. Dabei erweist sich das ACS als stabilster Lösungsansatz, welcher auch bei steigender Knotenanzahl stets eine Abweichung von unter 1,0% aufweist. Der GA eignet sich insbesondere für QAP mittlerer Größe und bestätigt damit den Eindruck, dass sich dieses Verfahren sehr gut als Werkzeug zur Optimierung logistischer Netzwerke eignet. Die verhältnismäßig hohen Abweichungen bei kleinen Problemstellungen sind auf die realen Anwendungen zurückzuführen. Wie in Abschnitt 4.4.4 beschrieben ist, weisen reale Problemstellungen insgesamt höhere Abweichungen auf, als theoretisch generierte Probleme. Zusammenfassend lässt sich sagen, dass auch bei diesem Vergleich der GA und das ACS am besten abschneiden.

5.3.3 Fazit

Auf Basis der durchgeführten Vergleiche und der Kategorisierung kann begründet dargestellt werden, welche Algorithmen zur Lösung welcher Problemstellung eingesetzt werden sollten. Bereits anhand der zur Verfügung stehenden Daten in dieser Ausarbeitung, ist ersichtlich in wieweit sich die Algorithmen für bestimmte Problemstellungen eignen. So ist es nicht verwunderlich, dass für die beiden ältesten Algorithmen, dem GA und den beiden Varianten der ACO, Daten für alle Optimierungsprobleme vorliegen. Alleine an der Anzahl an veröffentlichten Publikationen von Metaheuristiken zur Lösung des VRP wird deutlich, wie intensiv diese beiden Algorithmen für die vorliegenden Problemstellungen untersucht wurden. Insgesamt existierten im Jahr 2008 43 Publikationen zum GA, 10 für die ACO und insgesamt 51 Publikationen für alle weiteren Metaheuristiken in Bezug auf das VRP (Hanne und Dornberger 2017, S. 63), wobei die für diese Arbeit relevanten Algorithmen nur einen sehr kleinen Anteil ausmachen (Gendreau, et al. 2008). Zwar ist die Anzahl bis heute weiter gestiegen, doch hierdurch wird verdeutlicht, dass der GA und die ACO ein fester Bestandteil der kombinatorischen Optimierung sind. Dazu sei erwähnt, dass das MMAS nur für einfache Probleme eingesetzt werden sollte. Zwar erzielt das MMAS geringfügig bessere Lösungen als das ACS, doch benötigt dafür eine weitaus höhere Laufzeit (vgl. Abschnitt 4.2.1). Werden diese beiden Parameter gegenübergestellt, so ist eine Anwendung des ACS für komplexe und große Problemstellungen vorteilhafter.

Neben dem GA und der ACO erzielt der SFLA ebenfalls gute Ergebnisse. Obwohl dieser Algorithmus die jüngste der hier verglichenen Metaheuristiken ist, können durch die Anwendung des SFLA vereinzelt sogar die besten Ergebnisse erreicht werden. Allerdings ist der entscheidende Nachteil des SFLA seine fehlende Eignung auch für Standortprobleme Lösungen berechnen zu können. Der reine Fokus auf die Optimierung von Tourenproblemen hat zur Folge, dass der SFLA nicht so allgemeingültig wie der GA oder die ACO ist. Für einfache Problemstellungen liefern auch die diskreten Varianten der PSO gute Ergebnisse. Jedoch nimmt die Lösungsgüte ab, sobald das Problem komplexer und/oder größer wird. Insbesondere bei Problemen mit mehreren Zielfunktionen entfernen sich die erzielten Ergebnisse immer mehr vom Optimum. Für den HS Algorithmus lassen sich ähnliche Schlussfolgerungen ziehen. Für einfache und kleine Optimierungsprobleme erweist sich der HS Algorithmus in Bezug auf die Lösungsgüte als zuverlässige Methode zur Lösung von kombinatorischen Optimierungsproblemen. Mit zunehmender Problemkomplexität und -größe steigt auch die Abweichung zum Optimum, sodass der Einsatz anderer Algorithmen bessere Ergebnisse liefert.

Eine komprimierte Übersicht über die vom Autor empfohlenen Anwendungsgebiete der Algorithmen nach dem heutigen Stand der Technik ist in **Tabelle 5-1** dargestellt. Diese Einteilung erfolgt anhand von drei Kriterien. Das erste Kriterium sind die zur Verfügung stehenden Daten. Wie zuvor erwähnt liegen beispielsweise keine Daten des SFLA zur Lösung von Standortproblemen vor, sodass er für dieses Problem nicht als geeignet einzustufen ist. Desweiteren ist zumindest bei den Algorithmen der ACO eine Unterscheidung hinsichtlich

der Laufzeit möglich, sodass der bevorzugte Einsatz des ACS begründet werden kann. Das wichtigste Vergleichskriterium stellt jedoch die Lösungsgüte der Algorithmen dar. Im Rahmen dieser Arbeit wurde drauf geachtet, dass nur die besten Algorithmen für die jeweiligen Benchmarks in den Vergleich mit einbezogen wurden, sodass sicherstellt werden kann, dass diese Tabelle die Performance der Algorithmen hinsichtlich dieser Anwendungsgebiete nach dem heutigen Stand der Technik wiederspiegelt.

Tabelle 5-1: Empfohlene Anwendungsgebiete der Algorithmen zur Optimierung logistischer Netzwerke (Eigene Darstellung)

	Tourenplanungen						Standortplanungen					
	Einfach			Komplex			Einfach			Komplex		
Algorithmus	Klein	Mittel	Groß	Klein	Mittel	Groß	Klein	Mittel	Groß	Klein	Mittel	Groß
MMAS	✓	✓	✓	×	×	×	×	×	×	×	×	×
ACS	✓	✓	×	✓	✓	✓	×	×	×	✓	✓	✓
DPSO	✓	×	×	✓	✓	×	✓	✓	✓	×	✓	×
GA	✓	✓	✓	✓	✓	×	✓	✓	✓	×	✓	✓
SFLA	✓	✓	✓	✓	✓	✓	×	×	×	×	×	×
HS	✓	✓	×	×	×	×	✓	✓	✓	×	×	×

Anhand der gewonnen Erkenntnisse der vorliegenden Arbeit, empfiehlt der Autor folgende Anwendungen der Algorithmen. Es ist zu erkennen, dass keiner der vorliegenden Algorithmen als "der beste" Algorithmus angesehen werden kann. Zwar liefern der GA und das ACS am häufigsten die besten Ergebnisse, doch es existieren vereinzelte Problemstellungen, für die der SFLA oder die DPSO bessere Lösungen erzeugen. Diese Erkenntnis deckt sich mit dem sogenanntem "No-Free-Lunch" Theorem von Wolpert und Macready, welches besagt, dass kein universeller Algorithmus existiert, welcher für alle Optimierungsprobleme stets das beste Ergebnis liefern kann (Wolpert und Macready 1997, S. 77). Somit muss für jede Problemstellung der Einsatz eines bestimmten Algorithmus kritisch hinterfragt werden.

Mithilfe von den dargestellten Ergebnissen kann eine Vorauswahl von Algorithmen für ein breites Feld von Optimierungsproblemen in logistischen Netzwerken getroffen werden. Durch die veranschaulichte Bewertung der Algorithmen in **Tabelle 5-1** können anhand der Problemmerkmale die Algorithmen kategorisiert werden und bereits im Vorfeld ungeeignete Verfahren ausgeschlossen werden, ohne dass exakte Berechnungen notwendig sind.

6 Zusammenfassung und Ausblick

Das Ziel dieser Arbeit ist ein bewerteter Vergleich von fünf naturanalogen Algorithmen zur Optimierung logistischer Netzwerke. Der Ursprung der Algorithmen erstreckt sich dabei über ein weites Feld natürlicher Phänomene. So werden Algorithmen mit Ursprung in der biologischen Evolution, Tierschwärmen oder physikalischen Phänomenen betrachtet. Es werden Stärken und Schwächen dieser Algorithmen bei der Anwendung auf realitätsnahe Probleme hervorgehoben, sowie ihre Tauglichkeit für bestimmte Problemstellungen überprüft. Im Vordergrund stehen hierbei Herausforderungen zur Optimierung des Güterflusses zwischen Akteuren innerhalb eines logistischen Netzwerkes. Als weiteres Ziel wird auf mögliche Modifikationen der Algorithmen eingegangen und hervorgehoben inwieweit dies Auswirkungen auf die Lösungsgüte hat.

In der vorliegenden Arbeit werden zunächst alle betrachteten Algorithmen vorgestellt und die ursprünglichen Probleme, welche durch sie gelöst werden sollen, beschrieben. Zudem wird der historische Verlauf der Algorithmen erörtert und die Verbesserungen, welche im Laufe der Zeit an den Algorithmen vorgenommen wurden erklärt. Anschließend werden die Optimierungsprobleme definiert und ihre Bedeutung in logistischen Netzwerken hervorgehoben. Da der Güterfluss zwischen den Akteuren von essentieller Bedeutung für ein optimiertes logistisches Netzwerk ist, fokussiert sich diese Arbeit auf Routen-, Touren- und Standortprobleme. Für eine möglichst genaue Abbildung der Realität, beschränkt sich die Ausarbeitung auf diskrete Optimierungsprobleme.

Für jede der beschriebenen Problemstellungen werden charakteristische Benchmarks ausgewählt und die Lösungsgüte der Algorithmen hinsichtlich dieser Probleme überprüft. Dabei bezieht sich die Ausarbeitung auf die Untersuchungen anderer Autoren und stellt ihre Ergebnisse gegenüber. Um den Einfluss von Modifikationen zu untersuchen, werden sowohl Basisalgorithmen, als auch weiterentwickelte Algorithmen verglichen. Diese Arbeit verwendet jeweils die besten modifizierten Algorithmen, um eine genaue Übersicht zu erhalten, welche der vorliegenden Algorithmen nach dem heutigen Stand der Technik für Optimierungsprobleme geeignet sind und welche weitere Untersuchungen und Verbesserungen benötigen. Durch eine abschließende Kategorisierung, in der die Probleme nach bestimmten Charakteristiken gruppiert werden, ist eine qualitative Bewertung der Algorithmen möglich.

Aus dem Vergleich geht hervor, dass mit Ausnahme des SFLA, welcher nicht für Standortprobleme angewendet wird, alle Algorithmen grundsätzlich realisierbare Lösungen erzielen können. Für kleinere und einfache Problemstellungen erzielen fast alle Algorithmen ausreichend gute Lösungen. Zudem ist eine Korrelation zwischen dem *Alter* und der damit einhergehenden stetigen Verbesserung eines Algorithmus und der allgemeinen Lösungsgüte festzustellen. Die älteren und somit näher untersuchten und verbesserten Algorithmen, der GA und die ACO, erzielen problemübergreifend die stabilsten und besten Lösungen. Je komplexer und größer das Problem wird, umso stärker unterscheiden sich die Lösungen der

Algorithmen voneinander. Insbesondere das jüngste Verfahren, der HS Algorithmus, liefert im Vergleich mit den etablierten Metaheuristiken die schlechtesten Lösungen und bedarf somit weiterer Untersuchungen. Durch die vielen unterschiedlichen Formen der Diskretisierung, welche die Lösungsgüte beeinflussen, erweist sich die DPSO als zu unbeständiges Verfahren zur Lösung diskreter Optimierungsprobleme im Vergleich zu den weiteren Algorithmen. Diese Arbeit bestätigt das No-Free-Lunch-Theorem, sodass alle Algorithmen ihre Daseinsberechtigung für einzelne Problemstellungen besitzen. Aufgrund der vielfältigen Problemstellungen in logistischen Netzwerken, ist eine übersichtliche Einteilung der empfohlenen Anwendungsgebiete von Algorithmen umso wichtiger (vgl. **Tabelle 5-1**). Durch eine geeignete Wahl von Algorithmen zur Lösung von Optimierungsproblemen, können sowohl Zeit als auch Ressourcen gespart werden, da Berechnungen mit ineffizienten Algorithmen bereits im Vorfeld ausgeschlossen werden können.

Auf Grundlage dieser Ausarbeitung und insbesondere des Fazits in Abschnitt 5.3.3 lassen sich einige Aussagen über die Zukunft von naturanalogen Metaheurisiken zur Lösung von Optimierungsproblemen treffen. Es zeigt sich, dass Algorithmen kontinuierlich verbessert werden können, weshalb davon auszugehen ist, dass in Zukunft durch steigende Rechenkapazitäten und verbesserten Algorithmen immer größere und komplexere Optimierungsprobleme gelöst werden können. Insbesondere der SFLA und der HS Algorithmus liefern vielversprechende Ansätze, sodass sich die beiden Verfahren durch weitere Fortschritte zu zukunftsträchtigen Verfahren zur Optimierung in logistischen Netzwerken entwickeln können. Um auch die PSO als aussichtsvolle Methode zur Lösung von diskreten Optimierungsproblemen ansehen zu können, sind in Zukunft weitere Forschungen hinsichtlich der Art der Diskretisierung durchzuführen.

Literaturverzeichnis

Acker, Isabel Jasmin. 2011. *Methoden der mehrstufigen Ablaufplanung in der Halbleiterindustrie.* Hohenheim : Springer Gabler Verlag.

Amudha, Thangavel und Shivakumar, B. L. 2012. Improving the Solution Quality of Quadratic Assignment Problems using Harmony Improvised Consultant Guided Search Technique. *International Journal of Scientific and Engineering Research, Bd.3, 8.* S. 1300-1307.

Arnold, Dieter, Isermann, Heinz, Kuhn, Axel, Tempelmeier, Horst und Fuhrmans, Kai. 2007. *Handbuch Logistik, 3. Auflage.* Heidelberg : Springer Verlag.

Arnold-Rothmaier, Hildegard. 2016. www.econstor.eu. *Digitalisierung, Online-Handel und Smart Production: Chancen und Herausforderungen für die Logistikbranche.* [Online] 2016. [Zitat vom: 28. 07 2018.] https://www.econstor.eu/handle/10419/165796.

Auer, Benjamin, Holland, Heinrich, Kamps, Udo, Lübbecke, Marco und Rottmann, Horst. 2013. *Kompakt-Lexikon - Wirtschaftsmathematik und Statistik.* Wiesbaden : Springer Galber Verlag.

Beasly, John 2018. OR-Library. [Online] Februar 2018. [Zitat vom: 16. September 2018.] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/uncapinfo.html.

Berger, Jean, Barkaoui, Mohamed und Bräysy, Olli. 2003. A Route-Directed Hybrid Genetic Approach For The Vehicle Routing Problem With Time Windows. *Information System and Operational Research, Bd. 41, 2.* S. 179-194.

Bogon, Tjorben. 2012. *Agentenbasierte Schwarmintelligenz.* Trier : Springer Vieweg Verlag.

Boryczka, Urszula und Szwarc, Krysztof. 2018. The adaptation of the harmony search algorithm to the ATSP with the evaluation of the influence of the pitch adjustment place on the quality of results. *Journal of Information and Telecommunication.* S. 1-17.

Bouzidi, Morad und Riffi, Essaid Mohammed. 2014. Adaptation of the harmony search algorithm to solve the travelling salesman problem. *Journal of Theoretical and Applied Information Technology.* S. 154-160.

Bozorg-Haddad, Omid, Solgi, Mohammad und Loáiciga, Hugo A. 2017. *Meta-Heuristic and Evolutionary Algorithms for Engineering Optimization.* New Jersey : John Wiley & Sons, Inc.

Brunet, Frank Mcfarlane. 1959. *The clonal selection theory of acquired immunity.* Nashville : Vanderbilt University Press.

Burkard, Rainer. 2013. Quadratic Assignment Problems. [Buchverf.] Panos M. Pardalos, Ding-Zhu Du und Ronald L. Graham. *Handbook of Combinatorial Optimization, 2. Auflage.* New York : Springer. S. 2743-2805.

Burkard, Rainer, Cela, Eranda, Kanisch, Stefan, und Rendl, Franz. 2018. QAPLIB - A Quadratic Assignment Problem Library. [Online] Mai 2018. [Zitat vom: 17. September 2018.] http://anjos.mgi.polymtl.ca/qaplib//inst.html.

Buttelmann, Maik und Lohmann, Boris. 2004. Optimierung mit Genetischen Algorithmen und eine Anwendung zur Modellreduktion. *Automatisierungstechnik - Methoden und Anwendungen der Steuerungs-, Regelungs- und Informationstechnik, Bd.52, 4*. S. 151-163.

Chen, Chia-Ho und Ting, Ching-Jung. 2006. An improved Ant Colony System Algorithm for the Vehicle Routing Problem. *Journal of the Chinese Institute of Industrial Engineers, Bd.23, 2.* S. 115-126.

Chen, Shushu und Pu, Yifei. 2016. A Fractional-Order Weighted and Self-Adaptive Max-Min Ant System with 3-Opt Algorithm for Traveling Salesman Problem. *International Journal of Intelligent Information Systems, Bd.5, 4.* S. 48-54.

Cheng, Biyun, Lu, Haiyan, Huang, Yang und Xu, Kaibo. 2016. *An Improved Particle Swarm Optimization Algorithm Based on Cauchy Operator and 3-Opt for TSP.*Guangzhou : IEEE. Proceedings of 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT). S. 177-182.

Choi, In-Chan, Kim, Seong-In und Kim, Hak-Soo. 2003. A genetic algorithm with a mixed region search for the asymmetric traveling salesman problem. *Computers & Operations Research, Bd. 30, 1.* S. 773-786.

Clerc, Maurice. 2006. *Particle Swarm Optimization.* London : ISTE Ltd.

Cordón, Oscar, Herrera, Francisco und Thomas, Stützle. 2002. http://www.upc.edu/ca. *A review on the ant colony optimization metaheuristic: basis, models and new trends.* [Online] 2002. [Zitat vom: 13. 08 2018.] https://upcommons.upc.edu/bitstream/handle/2099/3624/1-cordon-herrera-stuetzle.pdf.

Danielsson, Per-Erik. 1980. Euclidean distance mapping. *Computer Graphics and Image Processing, Bd.14, 3.* S. 227-248.

Díaz, Bernabé Dorronsoro. 2006. The VRP Web. [Online] November 2006. [Zitat vom: 13. September 2018.] http://www.bernabe.dorronsoro.es/vrp/.

El-Abd, Mohammed, Hassan, Hassan und Kamel, Mohamed. 2009. *Discrete and Continuous Particle Swarm Optimization for FPGA.* Trondheim : IEEE. Proceedings of IEEE Congress on Evolutionary Computation. S. 706-711.

Dittmann, Lars. 2006. *Der angemessene Grad an Visibilität in Logistik-Netzwerken.* Wiesbaden : Deutscher Universitäts-Verlag.

Domschke, Wolfgang und Scholl, Armin. 2010. *Logistik: Rundreisen und Touren, 5. Auflage.* München : Oldenburg Wissenschaftsverlag.

Domschke, Wolfgang,Drexl, Andreas, Klein, Robert und Scholl, Armin. 2015. *Einführung in Operations Research, 9. Auflage.* Heidelberg : Springer Gabler Verlag.

Dorgio, Marco und Gambardella, Luca Maria. 1997. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *Transactions on Evolutionary Computation, Bd.1,1.* S. 53-66.

Dorgio, Marco und Stützle, Thomas. 2004. *Ant Colony Optimization.* London : MIT Press.

Dorgio, Marco, Maniezzo, Vittorio und Colorni, Alberto. 1996. Ant System: Optimization by a Colony of Cooperating Agents. *Proceedings of IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), Bd 26, 1.* S. 29-41.

Eberhart, Russell und Kennedy, James. 1995. *A new optimizer using particle swarm theory.*Nagoya : IEEE. Proceedings of the Sixth International Symposium on Micro Machine and Human Science. S. 39-43.

Eberhart, Russell und Shi, Yuhui. 2007. *Computational Intelligence - Concepts to Implementations.* Massachusetts : Morgan Kaufmann.

Elbeltagi, Emad, Hegazy, Tarek und Grierson, Donald. 2005. Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics,* Bd. 19, 1.

Eusuff, Muzaffar und Lansey, Kevin. 2003. Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm. *Journal of Water Resources Planning & Management, Bd.129, 3.* S. 210-225.

Fan, Huilian. 2010. Discrete Particle Swarm Optimization for TSP based on Neighborhood. *Journal of Computational Information Systems, Bd. 6,10.* S. 3407-3414.

Feldmann, Martin. 1999. *Naturanaloge Verfahren - Metaheuristiken zur Reihenfolgeplanung.* Wiesbaden : Deutscher Universitäts Verlag.

Fink, Andreas und Stefan, Voß. 2003. Anwendung von Metaheuristiken zur Lösung betrieblicher Planungsprobleme. *Wirtschaftsinformatik, Bd.45, 4.* S. 395-407.

Fisher, Len. 2010. *Schwarminzelligenz - Wie einfache Regeln Großes möglich machen.* Frankfurt am Main : Eichborn Verlag.

Focke, Axel. 2006. *Regionale Leistungs- und Krankenhausplanung - Ein Simulationsmodell auf Basis eines Ameisenalgorithmus.* Essen : Deutscher Universitäts Verlag.

Fogel, David. 2005. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, 3. Auflage.* New Jersey : John Wiley & Sons.

Gambardella, Luca Maria und Dorgio, Marco. 1996. *Solving Symmetric and Asymmetric TSPs by Ant Colonies.* Nagoya : IEEE. Proceedings of IEEE International Conference on Evolutionary Computation. S. 622-627.

Gambardella, Luca Maria, Taillard, Éric und Agazzi, Giovanni. 1999. MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. [Buchverf.] David Corne, Marco Dorgio und Fred Glover. *New ideas in optimization.* Maidenhead : McGraw-Hill Ltd.S. 63-76.

Gambardella, Luca Maria, Taillard, Éric und Dorgio, Marco. 1999. Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society, Bd.50, 2.* S. 167-176.

Geem, Zong Woo. 2010. State-of-the-Art in the Structure of Harmony Search. *Recent Advances in Harmony Search Algorithm.* Heidelberg : Springer Verlag. S. 1-10.

Geem, Zong Woo, Kim, Joong Hoon und Loganathan. 2001. A New Heuristic Optimization Algorithm: Harmony Search G.V. *SIMULATION: Transactions of The Society for Modeling and Simulation International, Bd.76, 2.* S. 60-68.

Gendreau, Michel, Potvin, Jean-Yves, Bräysy, Olli, Hasle, Geir und Lokketangen, Arne. 2008. Metaheuristics for the Vehicle Routing Problem and Its Extensions: A Categorized Bibliography. [Buchverf.] Bruce Golden, Raghu Raghavan und Edward Wasil. *The Vehicle Routing Problem: Latest Advances and new Challanges.* New York : Springer Verlag.S. 143-169.

Gerdes, Ingrid, Klawonn, Frank und Kruse, Rudolf. 2004. *Evolutionäre Algorithmen: Genetische Algorithmen - Strategein und Optimierungsverfahren - Beispielanwendungen.* Braunschweig : Friedr. Vieweg & Sohn Verlag.

Gomm, Moritz und Trumpfheller, Michael. 2004. Netzwerke in der Logistik. [Buchverf.] Hans-Christian Pfohl. *Netzkompetenz in Supply Chains.* Wiesbaden : Springer Gabler. S. 45-60.

Gong, Yue-Jiao, Zhang, Jun, Liu, Ou, Huang, Rui-Zhang, Chung, Henry Shu Hug und Shi, Yu-Hui. 2012. Optimizing the Vehicle Routing Problem With Time Windows: A Discrete Particle Swarm Optimization Approach. *Transaction on Systems, Man and Cybernetics - Part C: Applications and Reviews, Bd.42, 2.* S. 254-267.

Grötschel, Martin. 2007. Schnelle Rundreisen: Das Travelling-Salesman-Problem. [Buchverf.] Stephan Hußmann und Brigitte Lutz-Westphal. *Kombinatorische Optimierung erleben.* Wiesbaden : Friedr. Vieweg & Sohn Verlag. S. 95-129.

Guner, Ali R. und Sevkli, Mehmet. 2008. A Discrete Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem. *Journal of Artificial Evolution and Applications; Bd. 2008, 10.* S. 1-9.

Günes, Mesut, Spaniol, Otto und Kähmer, Martin. 2003. Der Ameisenroutingalgorithmus für mobile multi-hop Ad-hoc-Netze. *Praxis der Informationsverarbeitung und Kommunikation, Bd.26, 4.* S. 203-209.

Hafiz, Faizal und Abdennour, Adel. 2016. Particle Swarm Algorithm variants for the Quadratic Assignment Problems - A probabilistic learning approach. *Expert Systems with Applications, Bd.44, C.* S. 413-431.

Hanne, Thomas und Dornberger, Rolf. 2017. *Computational Intelligence in Logistics and Supply Chain Management.* Basel : Springer Verlag.

Heidari, Ali Asghar, Kazemizade, Omid und Abbaspour, Rahim A. 2015. *OBCHS: An effective Harmony Search Algorithm with Opposition Based Chaos-enhanced Initialization for Solving Uncapacitated Facility Location Problem.* Kish Island : ISPRS. Proceedings of International Conference onSensors & Modelsin RemoteSensing& Photogrammetry. S. 307-311.

Iordache, Serban. 2010. *Consultant-guided search: a new metaheuristic for combinatorial optimization problems.* New York : ACM. Proceedings of the 12th annual conference on Genetic and evolutionary computation. S. 225-232.

Jangura, Renu und Kait, Ramesh. 2017. *Analysis and Comparison Among Ant System; Ant Colony System and Max-Min Ant System With Different Parameters Setting.* Ghaziabad : IEEE. Proceedings of 3rd International Conference on Computational Intelligence & Communication Technology (CICT). S. 1-4.

Jaramillo, Jorge H., Bhadury, Joy und Batta, Rajan. 2002. On the use of genetic algorithms to solve location problems. *Computers & Operations Research, Bd. 29, 6.* S. 761-779.

Karakatič, Sašo und Podgorelec, Vili. 2015. A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing, Bd. 27, C.* S. 519-532.

Kennedy, James und Eberhart, Russell 2002. *A discrete binary version of the particle swarm algorithm.* Orlando : IEEE. Proceedings of IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation. S. 4104-4108.

Kennedy, James und Mendes, Rui. 2002 *Population Structure and Particle Swarm Performance.***.** Honolulu : IEEE. Proceedings of the Congress on Evolutionary Computation. S. 1671-1676.

Kennedy, James, Eberhart, Russell und Shi, Yuhui. 2001. *Swarm Intelligence.* San Francisco : Morgan Kaufmann Publishers.

Klaus, Peter, Krieger, Winfried und Krupp, Michael. 2012. *Gabler Lexikon Logistik, 5. Auflage.* Wiesbaden : Springer Gabler Verlag.

Kole, Arnab, Chakrabarti, Parichay und Bhattacharyya, Somnath. 2014. An Ant Colony Optimization Algorithm for Uncapacitated Facility Location Problem. *Artificial Intelligence and Applications, Bd.1, 1.* S. 55-61.

Li, Xiangyong, Tiang, Peng, Hua, Jing und Zhong, Ning. 2006. *A Hybrid Discrete Particle Swarm Optimization for the Traveling Salesman Problem.*Hefei : Springer Verlag. Proceedings of 6th International Conference SEAL. S. 181-188.

Lim, Meng-Hiot, Yuan, Yu und Omatu, Sigeru. 2002. Extensive Testing of a Hybrid Genetic Algorithm for Solving Quadratic Assignment Problems. *Computational Optimization and Applications, Bd. 23, 1.* S. 46-64.

Luo, Jianping und Chen, Min-Rong. 2014. Multi-phase modified shuffled frog leaping algorithm with extremal optimization for the MDVRP and the MDVRPTW. *Computers & Industrial Engineering, Bd. 72.* S. 84-97.

Luo, Jianping, Li, Xia, Cheng, Min-Rong und Liu, Hongwei. 2015. A novel hybrid shuffled frog leaping algorithm for vehicle routing problem with time windows. *Information Sciences, Bd.316, C.* S. 266-292.

Luo, Xue-hui, Yang, Ye und Li, Xia. 2008. Kaohsiung : IEEE. *Solving TSP with Shuffled Frog-Leaping Algorithm.* Proceedings of Eighth International Conference on Intelligent Systems Design and Applications. S. 228-232.

Maric, Miroslav. 2010. An efficient Genetic Algorithm for solvig the Multi-Level uncapacitated Facility Location Problem. *Computing and Informatics, Bd. 29, 2.* S. 183-201.

Marinakis, Yannis, Marinaki, Magdalene und Dounias, Georgios. 2010. A hybrid particle swarm optimization algorithm for the vehicle routing problem, Bd.23, 4. *Engineering Applications of Artificial Intelligence.* S. 463-472.

Matschiner, Markus und Zillmann, Mario. 2013. www.logistik-heute.de. *Branchendossier "Logistik und Transport".* [Online] 2013. [Zitat vom: 20. 07 2018.] https://www.logistik-heute.de/sites/default/files/logistik-heute/knowhow/branchendossier_logistik_und_transport_pdf__93358.pdf.

Melter, Robert. 1987. Some charaterizations of city block distance. *Pattern Recognition Letters, Bd.6, 4.* S. 235-240.

Moh'd Alia, Osama und Mandava, Rajeswari. 2011. The variants of the harmony search algorithm: an overview. *Artificial Intelligence Review, Bd.36,1.* S. 49-68.

Ming-Huwi, Hong. 2011. *Multilevel Image Thresholding by using the Shuffled Frog-leaping Optimization Algorithm.* Macao : IEEE. Proceedings of The 16th North-East Asia Symposium on Nano, Information Technology and Reliability. S. 144-149.

Näher, Stefan. 2006. www-i1.informatik.rwth-aachen.de/~algorithmus/index.php. *Das Travelling Salesman Problem- oder die optimale Tour für den Nikolaus.* [Online] 5. Dezember 2006. [Zitat vom: 4. August 2018.] www-i1.informatik.rwth-aachen.de/~algorithmus/algo40.php.

Naqvi, Najime Zehra und Matheru, Harmeen Kaur. 2011. *International Conference on Enviroment Sciene and Engineering IPCBEE, Bd.8.* Singapur: IACSIT Press. Proceedings of Review of Colony Optimization Algorithms on Vehicle Routing Problems and introduction to estimation-based ACO. S. 161-166.

Parsopoulos, Konstatinos und Vrahatis, Michael. 2004. A unified particle swarm optimization scheme. *LectureSeries on Computer and Computational Sciences, Bd.1.* S. 868-873.

Pfohl, Hans-Christian. 2004. *Logistiksysteme : betriebswirtschaftliche Grundlagen, 7. Auflage .* Berlin : Springer Verlag.

Pfohl, Hans-Christian. 2004. *Netzkompetenz in Supply Chains.* Wiesbaden : Springer Gabler Verlag.

Pfohl, Hans-Christian, Boldt, Oliver, Frunkze, Heiko, Gomm, Moritz, Hofmann, Erik, Trumpfheller, Michael und Sokolovsky, Veronika. 2004. Erfolgsfaktoren der Netzkompetenz in Supply Chains. [Buchverf.] Hans-Christian Pfohl. *Netzkompetenz in Supply Chains.* Wiesbaden : Springer Gabler Verlag. S. 141-171.

Pohlheim, Hartmut. 2000. *Evolutionäre Algorithmen - Verfahren, Operatoren und Hinweise für die Praxis.* Berlin Heidelberg : Springer Verlag.

Prins, Christian. 2004. A simple and effective evolutionary algorithm for the vehicle routing problem, Bd.31, 12. *Computers & Operations Research*. S. 1985-2002.

Reinelt, Gerhard. 1997. TSPLIB. [Online] 19. Februar 1997. [Zitat vom: 6. September 2018.] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html.

Reynolds, Craig. 2001. Boidseite von Craig Reynolds. [Online] 6. September 2001. [Zitat vom: 16. August 2018.] http://www.red3d.com/cwr/boids/.

Richard, Vahrenkamp. 2000. *Logistikmanagement, 4. Auflage.* München : Oldenbourg Verlag, 2000.

Rieck, Julia. 2008. *Tourenplanung mittelständischer Speditionsunternehmen.* Wiesbaden : Springer Gabler Verlag.

Saud, Suhair, Kodaz, Halife und Babaoglu, Ismail. 2017. *Solving Travelling Salesman Problem by using Optimization Algorithms.* Thonburi : KMUTT. Proceedings of The 9th International Conference on Advances in Information Technology. S. 13-32.

Shi, Xiaohu,Liang, Yanchun, Lee, Heowpueh und Lu, Chun. 2007. Particle swarm optimization-based algorithms for TSP and generalized TSP. *Information Processing Letters, Bd.103, 5.* S. 169-176.

Shtovba, Serhiy. 2005. Ant Algorithms: Theory and Applications. *Programming and Computer Software, Bd.31, 4.* S. 167-178.

Solomon, Marius M. 2005. VRPTW Benchmark Problems. [Online] 24. März 2005. [Zitat vom: 8. September 2018.] http://web.cba.neu.edu/%7Emsolomon/problems.htm.

Strasser, Shane, Goodeman, Rollie, Sheppard, John und Butcher, Stephyn. 2016 *Proceedings of the Genetic and Evolutionary Computation Conference.***.** Denver : ACM New York. A New Discrete Particle Swarm Optimization Algorithm. S. 53-60.

Stützle, Thomas und Hoos, Holger. 1997. *MAX-MIN Ant System and local search for the traveling salesman problem.* Indianapolis : IEEE. Proceedings of IEEE International Conference on Evolutionary Computation. S. 309-314.

Stützle, Thomas und Hoos, Holger. 2000. MAX-MIN Ant system. *Future Generation Computer Systems, Bd. 16,9.* S. 889-914.

Surekha, P. und Sumathi, S. 2011. Solution To Multi-Depot Vehicle Routing Problem Using Genetic Algorithms. *World Applied Programming, Bd.1, 3.* S. 118-131.

Sydow, Jörg. 1992. *Strategische Netzwerke: Evolution und Organisation.* Wiesbaden : Springer Gabler Verlag.

Walz, Guido. 2017. *Lexikon der Mathematik: Band 4.* Berlin : Springer Verlag.

Wang, Hong-bo, Zhang, Ke-peng und Tu, Xu-yan. 2015. A mnemonic shuffled frog leaping algorithm with cooperation and mutation. *Applied Intelligence, Bd. 43, 1.* S. 32-48.

Wang, Xiaolei, Gao, Xiao-Zhi und Zenger, Kai. 2015. *An Introduction to Harmony Search Optimization Method.* Heidelberg : Springer Verlag.

Weber, Jürgen, Bacher, Andreas und Groll, Marcus. 2002. Konzeption einer balanced scorecard für das controlling von unternehmensübergreifenden supply chains. *Controlling & Management, Bd.46, 3.* S. 133-141.

Weicker, Karsten. 2015. *Evolutionäre Algorithmen, 3. Auflage.* Wiesbaden : Springer Vieweg Verlag.

Wolpert, David H. und Macready, William G. 1997. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation, Bd.1, 1.* S. 67-82.

Xavier, Ivan. 2014. CVRPLIB. [Online] 2014. [Zitat vom: 09. September 2018.] http://vrp.atd-lab.inf.puc-rio.br/index.php/en/.

Yaseen, Saad Ghaleb und Nada M., A-Al-Samy. 2008. Ant Colony Optimization. *International Journal of Computer Science and Network Security, Bd. 8, 6.* S. 351-357.

Yassen, Esam, Ayob, Masri, nazri, Mohd Zarkee Ahmad und Ahmand, Zulkifli. 2013. Harmony Search Algorithm for Vehicle Routing Problem with Time Windows. *Journal of Applied Sciences, Bd. 13, 4.* S. 633-638.

Yu, B., Yang, Z. Z. und Xie, J. X. 2011. A parallel improved ant colony optimization for multi-depot vehicle routing problem. *Journal of the Operational Research Society, Bd. 62, 1.* S. 183-188.

Yuan, Zhenglei, Yang, Liliang, Wu, Haohua, Liao, Li und Li, Guoqiang. 2007. *Chaotic Particle Swarm Optimization Algorithm for Traveling Salesman Problem.* Jinan : IEEE Proceedings of IEEE International Conference on Automation and Logistics. S. 1121-1124.

Zhang, Changsheng, Sun, Jigui, Wang, Yan, Yang, Quingyun. 2007. *An Improved Discrete Particle Swarm Optimization Algorithm for TSP.* Silicon Valley : IEEE. Proceedings of International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops. S. 35-38.

Zhang, Jingmin und Wu, Congcong. 2012. Improved Shuffled Frog-Leaping Algorithm and Its Application. *Applied Mechanics and Materials, Bd. 155-156.* S. 92-96.

Zhao, Jia und Lv, Li. 2015. Two-Phases Learning Shuffled Frog Leaping Algorithm. *International Journal of Hybring Information Technology, Bd.8, 5.* S. 195-206.

Anhang

Anhang A: Nebenbedingungen der Problemstellungen

Traveling Salesman Problem (Domschke und Scholl 2010, S. 99-100)

$\sum_{j=1}^{n} x_{ij} = 1 \qquad für\ i = 1, \dots, n$ (6.1)

$\sum_{i=1}^{n} x_{ij} = 1 \qquad für\ j = 1, \dots, n$ (6.2)

$x_{ij} \in \{0, 1\} \qquad für\ i, j = 1, \dots, n$ (6.3)

$x_{i_1 i_2} + x_{i_2 i_3} + x_{i_k i_1} \leq k - 1 \ \forall\ Permutationen\ (i_1, i_2, \dots, i_k)\ und\ k = 2,3, \dots \frac{n}{2}$ (6.4)

Vehicle Routing Problem

Capacitated Vehicle Routing Problem (Rieck 2008, S. 13)

- Knotenmenge: $V = \{0, \dots, n\}$
- Fahrzeuganzahl: M
- Fahr- und Servicezeit: $f \geq 0$
- Servicezeit: $s_i \geq 0$
- Fahrzeit: $t_{ij} \geq 0$
- maximale Service- und Fahrzeit: $T_{max} \geq 0$
- Noch auszuliefernde Ladungsmenge: $lb \geq 0$
- Fahrzeugkapazität: Cap ≥ 0
- Kundenbedarf: $b_i \geq 0$

$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1 \qquad j \in V \setminus \{0\}$ (6.5)

$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ji} = 1 \qquad j \in V \setminus \{0\}$ (6.6)

$\sum_{i \in V\ \{0\}} x_{0i} \leq M$ (6.7)

$f_j \geq f_i + s_i + t_{ij} - T_{max}(1 - x_{ij}) \qquad i \in V, j \in V \backslash \{0\}$ (6.8)

$f_i + s_i + t_{i0} \leq T_{max} \qquad i \in V \backslash \{0\}$ (6.9)

$lb_i \geq lb_j + b_i - Cap(1 - x_{ij}) \qquad i \in V, j \in V\{0\}$ (6.9)

$b_i \leq lb_i \leq Cap \qquad i \in V$ (6.10)

$x_{ij} \in \{0, 1\} \qquad i, j \in V$ (6.11)

Vehicle Routing Problem with Time Windows (Rieck 2008, S. 18)

Zusätzlich zu den Nebenbedingungen (6.5) bis (6.11) kommen weitere Nebenbedingungen hinzu.

- Zeitfenster eines Kunden: $[a_i, b_i]\ mit\ a_i \leq b_i$
- Zeitfenster eines Depots: $[p, \varepsilon]$
  - Betriebsbeginn: p
  - Betriebsende: ε
- Anfangszeit der Bedienung in einem Knoten: τ

$$\tau_j \geq \tau_i + s_i + \tau_{ij} - (\varepsilon - p)\left(1 - x_{ij}\right) \quad i \in V, j \in V \setminus \{0\} \tag{6.12}$$

$$a_i \leq \tau_i \leq b_i \quad i \in V \setminus \{0\} \tag{6.13}$$

$$p \leq \tau_0 \tag{6.14}$$

$$\tau_i + s_i + t_{i0} \leq \varepsilon \quad i \in V \setminus \{0\} \tag{6.15}$$

Multi-Depot Vehicle Routing Problem (Surekha und Sumathi 2011, S. 120-121)

- Menge aller Depots: I → Depot Index i
- Menge aller Kunden: J → Kunden Index j
- Menge aller Fahrzeuge: K → Fahrzeugindex k
- Fahrzeuganzahl: N
- Distanz zwischen i und j: $d_{ij}\ mit\ i, j \in I \cup J$
- Maximaler Durchsatz eines Depots i: V_i
- Bedarf eines Kunden j: b_j
- Kapazität des Fahrzeugs k: cap_k
- Hilfsvariable zur Sub-Tour Vermeidung: U_{lk}
- Zusätzliche Entscheidungsvariable: $z_{ij} \begin{cases} 1 & wenn\ Kunde\ j\ Depot\ i\ zugeteilt \\ 0 & sonst \end{cases}$

$$\sum_{k \in K} \sum_{i \in I \cup J} x_{ijk} = 1 \quad j \in J \tag{6.16}$$

$$\sum_{j \in J} b_j \sum_{i \in I \cup J} x_{ijk} \leq cap_k \quad k \in K \tag{6.17}$$

$$U_{lk} - U_{jk} + N x_{ijk} \leq N - 1 \quad l, j \in J, k \in K \tag{6.18}$$

$$\sum_{j \in I \cup J} x_{ijk} - \sum_{j \in I \cup J} x_{jik} = 0 \quad k \in K, i \in I \cup J \tag{6.19}$$

$$\sum_{i \in I} \sum_{j \in J} x_{ijk} \leq 1 \quad k \in K \tag{6.20}$$

$$\sum_{j \in J} b_i z_{ij} \leq V_i \quad i \in I \tag{6.21}$$

$$-z_{ij} + \sum_{u \in I \cup J} \left(x_{iuk} + x_{ujk}\right) \leq 1 \quad i \in I, j \in J, k \in K \tag{6.22}$$

$x_{jik} \in \{0,1\}$ $i \in I, j \in J, k \in K$ (6.23)

$z_{ij} \in \{0,1\}$ $i \in I, j \in J$ (6.24)

$U_{lk} \geq 0$ $l \in J, k \in K$ (6.25)

Warehouse Location Problem

Einstufiges unkapazitiertes Warehouse Location Problem (Arnold, et al. 2007, S. 97)

- Anzahl der potentiellen Standorte: m
- Anzahl der Kunden/Akteure: n

$x_{ij} \leq y_i$ $f\ddot{u}r\ i = 1, \dots, m\ und\ j = 1, \dots, n$ (6.26)

$\sum_{i=1}^{m} x_{ij} = 1$ $f\ddot{u}r\ j = 1, \dots, n$ (6.27)

$y_i = \in \{0,1\}$ $f\ddot{u}r\ i = 1, \dots, m$ (6.28)

$x_{ij} \geq 0$ $\forall\ i\ und\ j$ (6.29)

Quadratic Assignment Problem (R. E. Burkard 2013)

- Menge der Betriebe: N
- Alle Permutationen φ der Menge N als n x n Matrix gegeben:

$$X = (x_{ij}) \text{ mit } x_{ij} = \begin{cases} 1 & wenn\ \varphi(i)\ j \\ 0 & sonst \end{cases}$$

$\sum_{i=1}^{n} x_{ij} = 1$ $j = 1, \dots, n$ (6.30)

$\sum_{j=1}^{n} x_{ij} = 1$ $i = 1, \dots, n$ (6.31)

$x_{ij} \in \{0,1\}$ $i, j = 1, \dots, n$ (6.32)

Anhang B: Parametersetting der Algorithmen

Vergleich 1: Symmetrisches TSP

- Für alle Tabellen gilt: n = Anzahl Knoten

Tabelle 6-1: Parametersetting für die erzielten Ergebnisse aus Tabelle 4-6

Algorithmus	Parameter	Wert
MMAS	Anzahl Ameisen	m=n oder m=n/2 mit n= Anzahl Knoten
	Einfluss der Pheromonmenge	α = 1
	Einfluss der bekannten Distanzen	β = 1
	Verdunstungsrate	p = 0,99
ACS	Anzahl Ameisen	m = 10
	Einfluss der Pheromonmenge	α = p
	Einfluss der bekannten Distanzen	β = 2
	Verdunstungsrate	p = 0,1
CPSO	Anzahl Partikel	Keine Angaben
	Gewichtungsfaktor "interna weight"	
	Kognitiver Einfluss	
	Sozialer Einfluss	
GA	Populationsgröße	s= 10 - 20
	Anzahl an Bewertung der Individuen	8 -128
	Anzahl an Generationen	100 - 1200
	Crossoverrate	p_c = 0,5
	Mutationsrate	p_m = 0,2
SFLA	Anzahl Memeplexes	m = 10
HS	Harmony Memory Consindering Rate	HMCR = 0,95
	Pitch Adjustment Rate	PAR = 0,45
	Größe der Harmony Memory	HM S= 10

Vergleich 2:Verhalten bei wachsender Problemstellung beim symmetrischen TSP

Tabelle 6-2: Parametersetting für die erzielten Ergebnisse aus Abbildung 4-2

Algorithmus	Parameter	Wert
MMAS	Anzahl Durchläufe	25
	Anzahl Ameisen	m=n oder m=n/2
	Einfluss der Pheromonmenge	α=1
	Einfluss der bekannten Distanzen	β=2
	Verdunstungsrate	p=0,98
	Iterationen	10.000
ACS	Anzahl Durchläufe	15
	Anzahl Ameisen	m=10
	Einfluss der Pheromonmenge	α=p
	Einfluss der bekannten Distanzen	β=2
	Wahrscheinlichkeitsparameter	q_0=0,9
	Verdunstungsrate	p=0,1
	Anzahl Bewertungen	10000*n
DPSO	Anzahl Durchläufe	100
	Anzahl Partikel	Keine Angaben
	Gewichtungsfaktor "interna weight"	
	Kognitiver Einfluss	
	Sozialer Einfluss	
	Iterationen	
GA	Anzahl Durchläufe	10 -20
	Populationsgröße	s=20
	Iterationen	50 - 1200
	Anzahl an Generationen	8 - 128
	Crossoverrate	p_c=0,5
	Mutationsrate	p_m=0,2
SFLA	Anzahl Durchläufe	50
	Anzahl Memeplexes	m=10
	Forschpoluation	10*n
	Submemeplex	2n/3
HS	Anzahl Durchläufe	10
	Harmony Memory Consindering Rate	HMCR= 0,95
	Pitch Adjustment Rate	PAR=0,45
	Größe der Harmony Memory	HMS= 10
	Iterationen	Keine Angaben

Vergleich 3: Asymmetrisches TSP

Tabelle 6-3: Parametersetting für die erzielten Ergebnisse aus Tabelle 4-7 und Abbildung 4-3

Algorithmus	Parameter	Wert
MMAS	Anzahl Durchläufe	25
	Anzahl Ameisen	m=n oder m=n/2
	Einfluss der Pheromonmenge	α=1
	Einfluss der bekannten Distanzen	β=2
	Verdunstungsrate	p=0,98
	Iterationen	20.000
ACS	Anzahl Durchläufe	15
	Anzahl Ameisen	m=10
	Einfluss der Pheromonmenge	α=p
	Einfluss der bekannten Distanzen	β=2
	Wahrscheinlichkeitsparameter	q_0=0,9
	Verdunstungsrate	p=0,1
	Anzahl Bewertungen	20000*n
DPSO	Anzahl Durchläufe	10
	Anzahl Partikel	N=25
	Gewichtungsfaktor "interna weight"	w=1
	Kognitiver Einfluss	c1=2
	Sozialer Einfluss	c2=2
	Größe der Nachbarschaftsliste	15
	Iterationen	2500*n
GA	Anzahl Durchläufe	3
	Populationsgröße	s=200
	Anzahl an Generationen	1000
	Crossoverrate	Keine Angaben
	Mutationsrate	
HS	Anzahl Durchläufe	30
	Harmony Memory Consindering Rate	HMCR= 0,98
	Pitch Adjustment Rate	PAR=0,25
	Größe der Harmony Memory	HMS= 5
	Iterationen	1000

Vergleich 4: CVRP

Tabelle 6-4: Parametersetting für die erzielten Ergebnisse aus Tabelle 4-8 und Abbildung 4-5

Algorithmus	Parameter	Wert
IACS	Anzahl Durchläufe	10
	Anzahl Ameisen	m=n/10
	Einfluss der Pheromonmenge	α=0,1
	Einfluss der bekannten Distanzen	β=4
	Wahrscheinlichkeitsparameter	q_0=0,5
	Verdunstungsrate	p=0,5
	Anzahl Iterationen	2*n
HybDPSO	Anzahl Durchläufe	Keine Angaben
	Anzahl Partikel	N=100
	Gewichtungsfaktor "interna weight"	Keine Angaben
	Iterationen	1000
GA	Anzahl Durchläufe	Keine Angaben
	Populationsgröße	30
	Anzahl an Generationen	Keine Angaben
	Crossoverrate	
	Mutationsrate	

Vergleich 5: VRTW

Tabelle 6-5: Parametersetting für die erzielten Ergebnisse aus Tabelle 4-9 und 4-10 und Abbildung 4-7, 4-8 und 4-9

Algorithmus	Parameter	Wert
MACS	Anzahl Durchläufe	3
	Anzahl Ameisen	m=10
	Einfluss der Pheromonmenge	α=0
	Einfluss der bekannten Distanzen	β=1
	Wahrscheinlichkeitsparameter	q_0=0,9
	Verdunstungsrate	p=0,1
S-PSO	Anzahl Durchläufe	30
	Anzahl Partikel	M=20
	Gewichtungsfaktor "interna weight"	Beginn: w=0,9 Ende: w=0,4
	Beschleunigungskoeffizient	c=2,0
	Wahrscheinlichkeit einer "Greedy-Initialisierung"	0,3
	Iterationen	1000-10.000
RHGA	Anzahl Durchläufe	3
	Populationsgröße	10 für Instanzen C1 und C2 50 für alle anderen
	Iterationen	Keine Angaben
	Anzahl an Generationen	
	Crossoverrate	
	Mutationsrate	
HSFLA	Anzahl Durchläufe	10
	Anzahl Memeplexes	m=8
	Forschpopulation	s=64
	Submemeplex	5
Meta-HS	Anzahl Durchläufe	10
	Harmony Memory Considering Rate	HMCR= 0,9
	Min Pitch Adjustment Rate	Par_{min}=0,3
	Max. Pitch Adjustment Rate	PAR_{max}=0,9
	Größe der Harmony Memory	HMS= 10
	Iterationen	Keine Angaben

Vergleich 6: MDVRP

Tabelle 6-6: Parametersetting für die erzielten Ergebnisse aus Tabelle 4-12 und Abbildung 4-10

Algorithmus	Parameter	Wert
PIACO	Anzahl Durchläufe	Keine Angaben
	Anzahl Ameisen	
	Einfluss der Pheromonmenge	
	Einfluss der bekannten Distanzen	
	Wahrscheinlichkeitsparameter	
	Verdunstungsrate	
GA	Anzahl Durchläufe	Keine Angaben
	Populationsgröße	50-100
	Iterationen	Problemabhängig
	Anzahl an Generationen	
	Crossoverrate	p_c=0,6
	Mutationsrate	p_m=0,02
MPMSFLA	Anzahl Durchläufe	10
	Forschpopulation	s=40
	Anzahl Memeplexes	Keine Angaben
	Submemeplex	

Vergleich 7: UWLP

Tabelle 6-7: Parametersetting für die erzielten Ergebnisse aus Abbildung 4-11 und 4-12

Algorithmus	Parameter	Wert
ACS	Anzahl Durchläufe	Keine Angaben
	Anzahl Ameisen	
	Einfluss der Pheromonmenge	α=0,9
	Einfluss der bekannten Distanzen	β=1
	Wahrscheinlichkeitsparameter	q0=0,9
	Verdunstungsrate	p=0,9
DPSOLS	Anzahl Durchläufe	30
	Anzahl Partikel	s=Anzahl Betriebe
	Kognitiver Einfluss	c1=0,5
	Sozialer Einfluss	c2=0,5
	Iterationen	250
GA	Anzahl Durchläufe	10
	Populationsgröße	s=n
	Iterationen	Keine Angaben
	Anzahl an Generationen	
	Crossoverrate	
	Mutationsrate	
OBCHS	Anzahl Durchläufe	30
	Harmony Memory Consindering Rate	Keine Angaben
	Pitch Adjustment Rate	
	Größe der Harmony Memory	
	Iterationen	

Vergleich 8: QAP

Tabelle 6-8: Parametersetting für die erzielten Ergebnisse aus Abbildung 4-13 und 4-14

Algorithmus	Parameter	Wert
HAS	Anzahl Durchläufe	10
	Anzahl Ameisen	m=10
	Einfluss der Pheromonmenge	α=0,1
	Einfluss der bekannten Distanzen	β=0,1
	Wahrscheinlichkeitsparameter	q0=0,9
	Verdunstungsrate	Keine Angaben
	Anzahl Bewertungen	
UPSO	Anzahl Durchläufe	10
	Anzahl Partikel	s=30
	Gewichtungsfaktor "interna weight"	w= 0,729
	Kognitiver Einfluss	c1=1,494
	Sozialer Einfluss	c2=1,494
	Vereinigungsfaktor	u=0,1
	Iterationen	Keine Angaben
PGA	Anzahl Durchläufe	10
	Populationsgröße	100
	Fitnessskalierungsvektor	Sf=3
	"zerofit-Schwelle"	Kz=5
	Konvergenzschwelle	ε=0,01%
	Max. Anzahl an Generationen	500
	Crossoverrate	Pc=0,8
	Mutationsrate	pm=0,05
HICGS	Anzahl Durchläufe	Keine Angaben
	Anzahl an Musikern	Problemabhängig
	Einfluss der angekündigten Kosten	β=0,002
	Max. Reputation	Rmax=40
	Reputation nach Sabbatical	Rstart=15
	Bonus für beste erreichte Reputation	bonus=6
	Abnehmrate der Reputation	rab=0,1
	Dauer des Sabbaticals	t=5
	Crossoverrate	rc=0,6
	Mutationsrate	rm=0,03
	Turniergröße	5
	Anzahl Variablen im Harmonievektor	Problemabhängig
	Min. Harmonieintervall	0
	Max. Harmonieintervall	4
	Bundweite	0,2
	Iterationen	1000

Anhang C: Zuordnung der Benchmarks

Tourenplanungen

Tabelle 6-9: Einteilung der Benchmarks für Tourenplanungen

Größe / Komplexität	Klein	Mittel	Groß
Einfach	burma14	eil51	kroA100
	bays29	berlin52	eil101
	p43	ft70	kro124
	ry48p	eil76	bier127
		pr76	ch130
			krob150
			ftv170
			d198
			ts225
			gil262
			att532
			rat778
Komplex	C1	C2	C3
	C6	C7	C12
	P01	P03	C8
	P02		c14
			C11
			C13
			C4
			C9
			C5
			C10
			R1
			R2
			C1
			C2
			RC1
			RC2
			P04
			P05

Standortplanungen

Tabelle 6-10: Einteilung der Benchmarks für Standortplanungen

Größe / Komplexität	Klein	Mittel	Groß
Einfach	Cap71		CapA
	Cap72		CapB
	Cap73		CapC
	Cap101		
	Cap102		
	Cap103		
	Cap104		
	Cap131		
	Cap132		
	Cap133		
	Cap134		
Komplex	tai20a	sko56	tai100a
	tai20b	tai60a	tai100b
	chr25a	tai60b	tai150b
	bur26a	esc64a	tho150
	nug30	sko72	wil150
	tai30a	tai80a	
	tai30b	tai80b	
	kra30a	sko81	
	kra30b	sko90	
	ste36a		
	tai40a		
	tai40b		
	sko42		
	tai50a		