Technische Universität München
Lehrstuhl für Computergestützte Modellierung und Simulation
Fakultät für Bauingenieur- und Vermessungswesen

Optimierung der Instandsetzungszeitpläne städtischer Infrastrukturbauwerke auf Basis von Metaheuristiken

Katharina Lukas

Vollständiger Abdruck der von der Fakultät für Bauingenieur- und Vermessungswesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr.-Ing. Chr. Gehlen
Prüfer der Dissertation:	1. Univ.-Prof. Dr.-Ing. A. Borrmann
	2. Univ.-Prof. Dr.-Ing. R. Gerike, Universität für Bodenkultur Wien, Österreich
	3. Priv-Doz Dr.-Ing. habil. V. Berkhahn, Leibniz Universität Hannover

Die Dissertation wurde am 15.11.2012 bei der Technischen Universität München eingereicht und durch die Fakultät für Bauingenieur- und Vermessungswesen am 05.04.2013 angenommen.

Zusammenfassung

Um Sicherheit und Gebrauchstauglichkeit von Bauwerken über ihre Lebensdauer gewährleisten zu können, sind regelmäßige Instandsetzungsarbeiten notwendig. Bei Infrastrukturbauwerken, gerade in feinmaschigen Straßennetzen, wie sie vor allem im städtischen Raum auftreten, bedeuten diese Instandsetzungsarbeiten immer auch eine Störung des Straßenverkehrs. Um diese Störungen so gering wie möglich zu halten, ist eine Betrachtung des gesamten Netzes, nicht nur der einzelnen Bauwerke erforderlich. Ziel dieser Arbeit war die Entwicklung von Optimierungsstrategien für Instandsetzungszeitpläne für so geartete Szenarien. Dazu wurde das Problem zunächst formal definiert. Beschrieben wurden dabei drei verschiedene Zielkriterien, die einzeln aber auch kombiniert als mehrkriterielles Problem optimiert werden. Eine Erweiterung um weitere Zielkriterien ist denkbar. Als mögliche Lösungsstrategien wurden die Metaheuristiken Genetische Algorithmen und Ameisenalgorithmen auf das Problem adapiert und ihre Eignung auf mehreren Testszenarien untersucht.

Abstract

To ensure the safety and serviceability of structures over their entire lifetime, maintenance work on a regular basis is essential. For infrastructural assets, mainly in fine-meshed road networks, as they occur mainly in urban areas, these maintenance measures always result in a disturbance of road traffic. To keep these disturbances as low as possible, it is necessary to consider not the single structures but the entire network. The aim of this thesis was the developement of optimization strategies for this type of maintenance schedules. For this firstly the problem was formally defined. Three different objectives were defined, that can be optimized singly or combined as a multi-criteria optimization problem. As possible solution strategies the meta-heuristics Genetic Algorithms and Ant Colony Optimization were adapted to the problem and tested for their usability on several test scenarios.

Vorwort

Diese Arbeit entstand während meiner Tätigkeit als Wissenschaftliche Mitarbeiterin am Lehrstuhl Computation in Engineering und ab Februar 2011 am damals neugegründeten Fachgebiet, seit Februar 2012 Lehrstuhl, für Computergestützte Modellierung und Simulation an der Technischen Universität München.

An dieser Stelle möchte ich allen denen danken, die zum Gelingen dieser Arbeit beigetragen haben. Zuvorderst danken möchte ich meinem Doktorvater Prof. Dr.-Ing. André Borrmann. Ich weiß die vielen fruchtbaren Diskussionen, die guten Ratschläge und konstruktiven Hinweise sehr zu schätzen.

Danken möchte ich auch meinem „Doktorgroßvater“ Prof. Dr. rer. nat. Ernst Rank für die Aufnahme an seinem Lehrstuhl und für seine konstruktiven Anregungen.

Frau Prof. Dr.-Ing. Regine Gerike und Herrn PD Dr.-Ing. habil. Volker Berkhahn danke ich für die Übernahme des Koreferrates sowie einige wichtige Hinweise, die entscheidend zu dieser Arbeit beigetragen haben.

Für die fast familiäre Atmosphäre an beiden Lehrstühlen möchte ich mich bei allen derzeitigen und ehemaligen Kollegen von CiE und CMS bedanken. Mein Dank gilt dabei auch der Rechnerbetriebsgruppe, insbesondere Markus Teßmann und Robert Dietl, die es mir ermöglichten, einen der CIP-Pools an mehreren Wochenenden für meine Rechnungen zu nutzen.

Der Firma PTV AG danke ich dafür, dass sie mir eine Lizenz des Verkehssimulators VISUM zur Verfügung stellten. Meinem Bachelorstudenten und späterem Hiwi Alexander Schraa danke ich für die eifrige Erstellung von Testszenarien.

Herrn Friedrich-Wilhelm Menke vom Baureferat der Landeshauptstadt München danke ich für eine Reihe von aufschlussreichen Gesprächen zum derzeitigen praktischen Vorgehen in der Instandsetzungsplanung.

Mein Dank gilt auch meinen Eltern, die mir mein Studium ermöglichten und mir bei meinem Bildungsweg immer den Rücken stärkten und mich ermutigten meinen eigenen Weg zu gehen.

Ebenfalls ein herzlicher Dank an meinen Partner Dr. Dirk Wünderlich, der mir in den letzten Jahren Kraft gegeben hat, mich aufgebaut hat, wenn die Dinge nicht so liefen wie gewünscht und mir auch sonst mit Ratschlägen zur Seite stand.

Schließlich noch mein Dank an Fabian Ritter und Andrea Zimmer, die diese Arbeit Korrektur gelesen und durch viele Hinweise verbessert haben.

Inhaltsverzeichnis

1. Einführung **1**
1.1. Motivation und Zielsetzung 1
1.2. Aufbau der Arbeit 2

2. Optimierungsprobleme **5**
2.1. Einkriterielle Optimierung 5
2.2. Kombinatorische Optimierung 8
2.3. NP-harte Probleme 11
2.4. Lösungsstrategien für NP-harte Probleme 12
2.4.1. Greedy-Algorithmus 13
2.4.2. Bergsteigeralgorithmus 16
2.4.3. Tabu-Suche 17
2.4.4. Simuliertes Abkühlen 18
2.4.5. Populationsbasierte Ansätze 19
2.5. Mehrkriterielle Optimierung 19
2.5.1. Allgemeines Vorgehen 19
2.5.2. Dominanz und Pareto-Optimalität 21

3. Optimierung im Infrastrukturmanagement **25**
3.1. Konventionelles Vorgehen beim Infrastrukturmanagement 25
3.2. Verwandte Arbeiten 26
3.3. Problembeschreibung 30
3.3.1. Zielfunktionen 33
3.3.2. Randbedingungen 36
3.3.3. Zusammenfassung 38

4. Eingangsgrößen **39**
4.1. Lebensdauermanagement auf Bauwerksebene 39
4.1.1. Schäden an Bauwerken 39
4.1.2. Bauwerksmanagement nach DIN 1076 41
4.1.3. Ergänzende Ansätze zum Bauwerksmanagement 44
4.1.4. Prototypische Umsetzung eines prädiktiven Lebensdauermanagementsystems 47
4.1.5. Zusammenfassung 54
4.2. Modellierung von Straßenverkehr 55
4.2.1. Entstehung von Verkehr 55
4.2.2. Kategorisierung von Verkehrsmodellen 55

4.2.3. Modellbildung . . . 56
4.2.4. Simulation . . . 57
4.2.5. Der Vier-Stufen-Algorithmus . . . 58
4.2.6. Zusammenfassung . . . 62

5. Genetische Algorithmen 65
5.1. Grundlagen . . . 65
5.2. Algorithmus . . . 65
5.2.1. Lösungsrepräsentation . . . 66
5.2.2. Ablauf der Optimierung . . . 67
5.2.3. Behandlung ungültiger Lösungen . . . 75
5.3. Genetische Algorithmen für das Lebensdauermanagement . . . 79
5.3.1. Mögliche Repräsentationen . . . 79
5.3.2. Kreuzungs- und Mutationsoperatoren . . . 82
5.3.3. Behandlung ungültiger Lösungen . . . 86
5.3.4. Zusammenfassung . . . 94
5.4. Genetische Algorithmen für mehrkriterielle Probleme . . . 95
5.4.1. Non-Generational Genetic Algorithm for Multi-objective Optimization . . . 97
5.4.2. Ansatz von Jaszkiewicz . . . 101
5.4.3. Zusätzliche Ergänzungen für das mehrkriterielle Instandsetzungsproblem . . . 105
5.4.4. Zusammenfassung . . . 106

6. Ameisenalgorithmen 107
6.1. Grundlagen . . . 107
6.2. Algorithmus . . . 109
6.2.1. Ant System . . . 109
6.2.2. Elitist Ant System . . . 113
6.2.3. Rank-Based Ant System . . . 115
6.2.4. Ant Colony System . . . 115
6.3. Ameisenalgorithmen für das Lebensdauermanagement . . . 119
6.3.1. Formulierung des Problems . . . 119
6.3.2. Heuristische Information η . . . 122
6.3.3. Dynamische Anpassung von β . . . 124
6.3.4. Behandlung ungültiger Lösungen . . . 125
6.3.5. Lokale Suche . . . 129
6.3.6. Zusammenfassung . . . 133
6.4. Ameisenalgorithmen für mehrkriterielle Probleme . . . 133
6.4.1. Bi-Criterion Ant . . . 134
6.4.2. Pareto Ant Colony Optimization . . . 136
6.4.3. Zusammenfassung . . . 138

7. Ergebnisse 139
7.1. Testszenarien . . . 139

7.2. Einkriterielles Problem 144
7.2.1. Genetische Algorithmen 144
7.2.2. Ameisenalgorithmen 151
7.2.3. Zusammenfassung 158
7.3. Mehrkriterielles Problem 159
7.3.1. Genetische Algorithmen 161
7.3.2. Ameisenalgorithmen 168
7.3.3. Zusammenfassung 169

8. Zusammenfassung und Ausblick **173**

A. Testszenarien **177**
A.1. Szenario 1 177
A.2. Szenario 2 180
A.3. Szenario 3 187
A.4. Szenario 4 194

1. Einführung

1.1. Motivation und Zielsetzung

Infrastrukturbauwerke wie Brücken, Tunnel aber auch Stützwände, Unterführungen und ähnliches bilden eine wertvolle Grundlage für wirtschaftliches Wachstum und Stabilität. Gleichzeitig enthalten diese Bauwerke aber auch ein hohes Gefährdungspotential, wie sich immer wieder bei Brückeneinstürzen zeigt (Brückenweb, 2012). Eine häufige Ursache für derartige Katastrophen ist mangelhafte Wartung. Da die Bauwerke schädlichen Umgebungsbedingungen ausgesetzt sind, unterliegen sie einem Alterungsprozess. Um ihre Sicherheit dennoch zu gewährleisten, sind regelmäßige Instandsetzungsarbeiten nötig.

In Deutschland sind, wie in einem Großteil der Industrienationen, Art, Umfang und Häufigkeit der Bauwerksüberwachungen durch Gesetze und Regelwerke geregelt. Zusätzlich fand in den letzten Jahren eine verstärkte Forschungstätigkeit in Richtung besserer Prognosemodelle zum Alterungsprozess von Bauwerken statt.

Da Bauwerksbetreiber in der Regel nicht nur ein einzelnes Bauwerk sondern eine größere Menge von Bauwerken verwaltet, genügt es nicht, Instandsetzungs- und Wartungsarbeiten nur auf Bauwerksebene zu planen. Begrenzte Ressourcen machen es notwendig, auf Netzwerkebene zu planen um so die Sicherheit für alle Bauwerke sicherzustellen ohne z.B. Budgetgrenzen zu überschreiten.

Auch für die Fragestellung der optimalen Budgetverteilung existieren bereits Lösungsansätze. Diese betrachten jedoch nur Bauwerke an Überlandstraßen und Autobahnen. Für Betreiber in städtischen Netzen kommen neben der rein monetären Betrachtung jedoch noch andere Fragestellungen dazu. So ist in den feinmaschigen städtischen Netzen der Einfluss von Baustellen auf den Straßenverkehr wesentlich höher, da sich die einzelnen Störungen leichter überlagern können. Dazu kommt, dass in derartigen Netzen die gleichzeitige (Teil-) Sperrung mehrerer Straßen durch gegenseitige Beeinflussung der Verkehr wesentlich stärker behindert werden kann, als durch die Summe der Einzelsperrungen. Auch andere Fragestellungen treten vornehmlich in städtischen Netzen auf. Beispielsweise werden städtische Infrastrukturbauwerke häufig auch von dritter Seite genutzt, zum Beispiel von Anbietern für öffentlichen Verkehr oder Telefonanbietern. Wechselwirkungen mit deren Bedürfnissen sind ebenfalls zu berücksichtigen.

Ziel dieser Arbeit ist die Entwicklung von Optimierungsmethoden für die Instandsetzungsplanung für Infrastrukturbauwerke in städtischen Netzen. Die Bauwerke können dabei innerhalb ihrer Instandsetzungsfrist frei verschoben werden, wobei Randbedingungen durch Budget und verfügbare Ressourcen die Anzahl der pro Jahr instand zu setzenden Bauwerke festlegen. Durch die so entstehenden unterschiedlichen Kombinationen von gleichzeitigen Instansetzungen wird der Straßenverkehr stärker oder weniger stark gestört. Ein besonderer Schwerpunkt dieser Arbeit liegt auf der Minimierung dieser Störung des Straßenverkehrs, die von den Instandsetzungsarbeiten ausgeht. Es sollen jedoch auch mehrkriterielle Ansätze untersucht werden, die es erlauben, mehrere Optimierungsziele gleichzeitig zu betrachten.

Da es sich bei diesen Problemen um NP-harte Probleme handelt, bieten sich zu deren Lösung vor allem metaheuristische Ansätze an. In dieser Arbeit wurden dazu Genetische Algorithmen und Ameisenalgorithmen ausgewählt. Beide Ansätze werden auf ihre grundsätzliche Anwendbarkeit auf die genannte Problemstellung untersucht und im Hinblick auf ihre Eignung für eine Anwendung in der Praxis miteinander verglichen. Zur Modellierung des Verkehrs wurden die Optimierungsalgorithmen mit einem Verkehrssimulator gekoppelt um die Zielfunktion der Verkehrsbeeinflussung auszuwerten. Da diese Auswertung verhältnismäßig zeitintensiv ist, wude besonders großer Wert darauf gelegt, solche Methoden zu finden, die mit möglichst wenig Zielfunktionsaufrufen dennoch gute Lösungen finden, um das entwickelte System möglichst praxistauglich zu gestalten.

1.2. Aufbau der Arbeit

Die Arbeit ist wie folgt aufgebaut:

Kapitel 2 beschreibt allgemeine Grundlagen zur Optimierung im ein- und mehrkriteriellen Fall. Dabei wird besonders auf kombinatorische Probleme eingegangen. Zusätzlich werden einige klassische kombinatorische Optimierungsprobleme und grundsätzliche Lösungsstrategien vorgestellt.

Kapitel 3 beschreibt zunächst das derzeit übliche Vorgehen im Infrastrukturmanagement. Darauf folgt ein Überblick über Arbeiten, die sich ebenfalls mit Instandsetzungszeitplänen für verschiedene Anwendungsbereiche befassen. Das Kapitel schließt mit einer Definition des in dieser Arbeit behandelten Optimierungsproblems mit seinen Zielfunktionen und Randbedingungen.

In **Kapitel 4** werden die externen Eingangsgrößen der Optimierung behandelt. Dazu gliedert es sich in zwei Teile: Der erste Teil beschäftigt sich mit den Schädigungsmechanismen an Bauwerken sowie dem Infrastrukturmanagement auf Ebene eines einzelnen Bauwerks. Im zweiten Teil wird der Straßenverkehr betrachtet. Es wird darauf eingegangen, wie Verkehr entsteht, wie er modelliert werden kann und wie er als Zielfunktion in der Optimierung Eingang findet.

Die **Kapitel 5** und **6** bilden den Kern der Arbeit. Hier werden die verwendeten Algorithmen beschrieben und es wird darauf eingegangen, wie sie zur Behandlung des Instandsetzungsproblems adaptiert werden müssen. In Kapitel 5 geschieht dies für die Genetischen Algorithmen, in Kapitel 6 für die Ameisenalgorithmen.

In **Kapitel 7** erfolgt dann ein Vergleich und eine Bewertung der untersuchten Algorithmen anhand von Testläufen auf verschiedenen Szenarien.

Kapitel 8 rundet die Arbeit durch eine Zusammenfassung der Ergebnisse ab und liefert einen Ausblick auf zukünftige Forschungsaufgaben.

2. Optimierungsprobleme

2.1. Einkriterielle Optimierung

Ziel einer einkriteriellen Optimierung ist es, für ein Problem aus der Menge aller möglichen Lösungen, dem Lösungsraum bzw. Suchraum X, die optimale Lösung[1] zu finden.

Bei dieser optimalen Lösung handelt es sich, je nach Problemstellung, um ein Maximum oder Minimum der Zielfunktion $f : X \to \Re$, die das Zielkriterium des Problems beschreibt. Handelt es sich um die Suche nach dem Maximum, so spricht man von einem Maximierungsproblem, ist dagegen das Minimum gesucht, so spricht man von einem Minimierungsproblem. Durch Umkehr des Vorzeichens der Zielfunktion lassen sich Maximierungsprobleme leicht in Minimierungsprobleme überführen und umgekehrt (Dualitätspinzip, vgl. z.B. Deb (2004)).

Bei einem Optimum kann es sich um ein lokales oder um ein globales Optimum handeln[2]. Zu deren Unterscheidung ist es zunächst notwendig, den Begriff *Nachbarschaft* zu definieren: Betrachtet man den Lösungsraum X als eine n-dimensionale Punktmenge [3], so ist die Nachbarschaft N einer Lösung x, beschrieben durch einen Punkt in dieser Menge, die Menge aller anderen Lösungen, die in einem beliebig kleinen Intervall um x liegen (vgl. hierzu auch Michalewicz & Fogel (2004)).

Eine Lösung $\hat{x} \in X$ bezeichnet man als lokales Maximum, wenn für die Zielfunktion f gilt $f(\hat{x}) \geq f(x)$ für alle Lösungen x in der Nachbarschaft von $\hat{x}$. Die Definition für ein lokales Minimum $\check{x} \in X$ ist entsprechend: $f(\check{x}) \leq f(x)$ für alle Lösungen x in der Nachbarschaft von $\check{x}$. Ein lokales Optimum ist ein lokales Maximum für ein Maximierungs- bzw. ein lokales Minimum für ein Minimierungsproblem.

Für ein globales Maximum $\hat{x}$ gilt dagegen $f(\hat{x}) \geq f(x) \forall x \in X$, bzw. für ein globales Minimum $\check{x}$: $f(\check{x}) \leq f(x) \forall x \in X$. Ein globales Optimum ist ein globales Maximum für ein Maximierungs- bzw. ein globales Minimum für ein Minimierungsproblem. Das globale Optimum ist also immer auch ein lokales Optimum. Andersherum kann ein lokales Optimum einen wesentlich schlechteren Zielfunk-

[1] bzw. eine finite Menge optimaler Lösunge in dem Fall, dass mehrere Lösungen mit dem gleichen optimalen Wert vorhanden sind

[2] Zur Definition von lokalen und globalen Optima siehe auch Weise (2009)

[3] Für ein Optimierungsproblem mit n Parametern

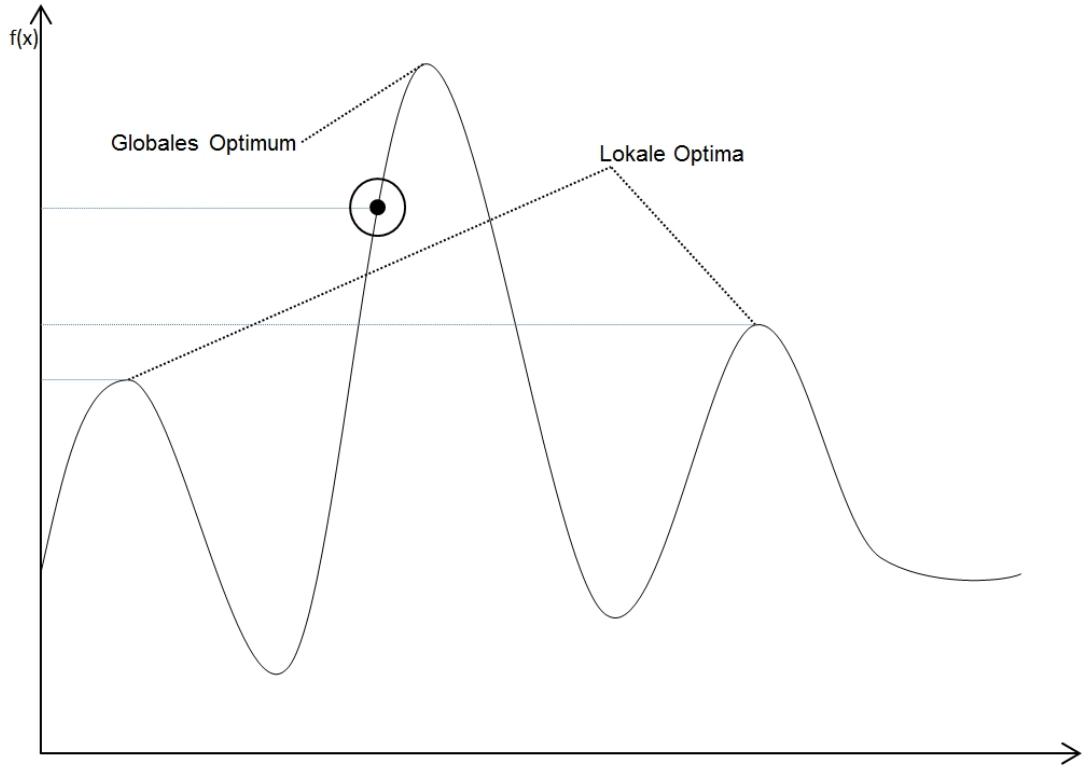

Abbildung 2.1.: Lokale Optima und globales Optimum einer eindimensionalen Funktion $f(x)$ (Maximierung). Die markierte Lösung hat einen höheren Funktionswert als die beiden lokalen Maxima, ist aber selber kein Maximum, da in ihrer Nachbarschaft (gekennzeichnet durch den Kreis) Lösungen mit höherem Funktionswert existieren.

tionswert haben als das globale Optimum. Es kann auch vorkommen, dass eine Lösung, die kein (lokales oder globales) Optimum ist, einen besseren Funktionswert hat, als ein lokales Optimum, da zu dessen Definition nur seine unmittelbare Nachbarschaft herangezogen wird (vgl. Abbildung 2.1).

Diese Definitionen gelten für den Fall, dass der gesamte Lösungsraum zulässig ist. Häufig ist die Suche jedoch durch zusätzliche Randbedingungen eingeschränkt. Diese Randbedingungen werden durch Gleichungen und Ungleichungen der Lösungsparameter beschrieben, die für die zulässigen Lösungen des Problems erfüllt sein müssen. Die zulässigen Lösungen bilden zusammen den gültigen Lösungsraum $F \subseteq X$. Dieser muss nicht notwendigerweise zusammen hängen. Dies bedeutet, um von einer gültigen Lösung rein über Nachbarschaftsbeziehungen zu einer anderen gültigen Lösung zu gelangen, kann es sein, dass ungültige Lösungen als Zwischenschritte benutzt werden müssen (vgl. Abbildung 2.2).

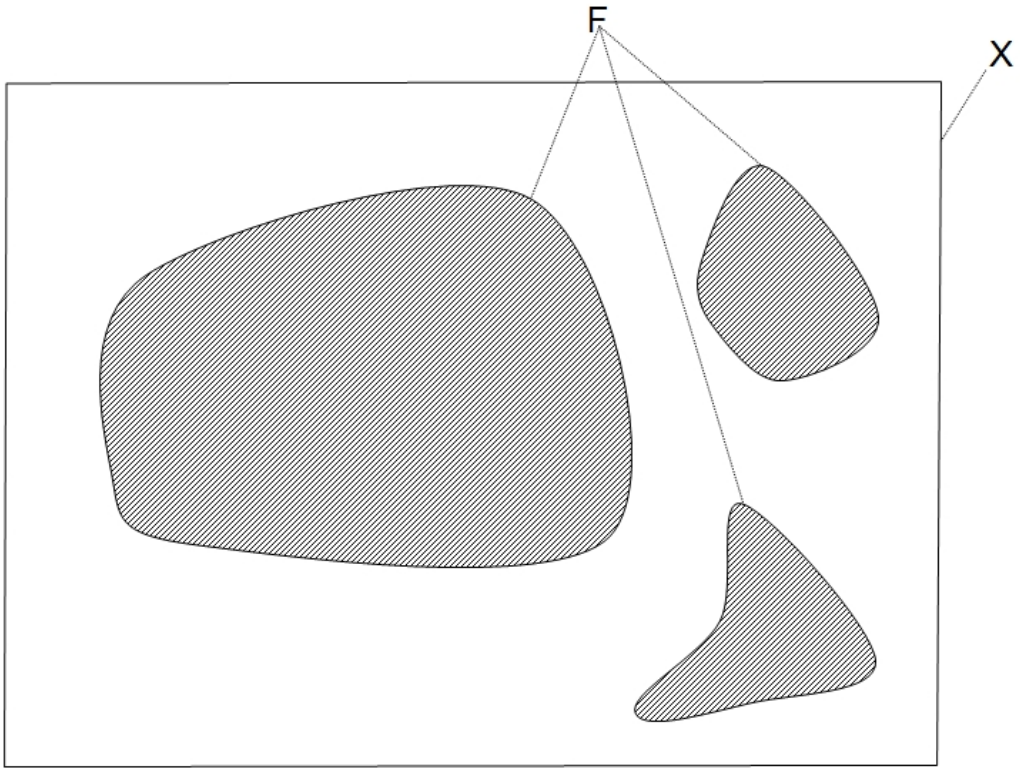

Abbildung 2.2.: Lösungraum X und gültiger Lösungsraum F im zweidimensionalen Raum. Der gültige Bereich des Lösungsraumes F hängt nicht notwendigerweise zusammen (Abbildung modifiziert nach (Michalewicz & Fogel, 2004)).

2.2. Kombinatorische Optimierung

Eine Unterteilungsmöglichkeit für Optimierungsprobleme ist die Unterscheidung, ob der Lösungsraum X kontinuierlich oder diskret ist. Probleme mit diskretem Lösungsraum bezeichnet man auch als *kombinatorische Optimierungsprobleme*.

Bei dem Lösungsraum eines kombinatorischen Optimierungsproblems handelt es sich also um eine finite oder abzählbar unendliche Menge. Häufig vorkommende mögliche Beschreibungen für die Lösungen eines solchen Problems (also die Objekte im Lösungsraum) sind Tupel, Mengen, Permutationen oder Graphen.

Ein typisches Beispiel für ein kombinatorisches Optimierungsproblem ist das Problem des Handlungsreisenden (engl. *Travelling Salesperson Problem*, TSP). Da es im weiteren Verlauf dieser Arbeit noch öfter als Beispiel dienen wird, sei es hier kurz beschrieben.

Intuitiv geht es beim TSP um die Suche der besten (kürzesten) Route für einen Vertreter, der Kunden in mehreren Städten besuchen und danach zu seinem Ausgangspunkt zurück kehren muss[4] (für ein Beispiel siehe Abbildung 2.3). Mathematisch betrachtet handelt es sich um die Suche nach dem kürzesten Hamilton-Kreis[5] in einem vollständigen Graph[6] mit gewichteten Kanten. Die Knoten des Graphen entsprechen dabei den Städten, die Kantengewichte den Abständen (je nach Problemstellung z.B. euklidisch, nach Manhatten-Metrik oder als Reisezeit angegeben) zwischen den entsprechenden Städten. Sind für alle Städte i, j die Abstände ij und ji gleich, dann spricht man von einem symmetrischen TSP, sonst von einem assymetrischen.

Ein weiteres wichtiges kombinatorisches Optimierungsproblem ist das Rucksackproblem sowie dessen Ableger, das Multiple Rucksackproblem. Diese Probleme sind hier von Bedeutung, da sie sehr eng mit dem in dieser Arbeit behandelten Problem verwandt sind. Ausgangspunkt des Rucksackproblems ist eine Liste von Gegenständen mit jeweils einem Nutzwert und einem Gewicht. Ziel ist es nun, aus dieser Liste eine Menge von Gegenständen so auszuwählen, dass ihr Nutzen maximiert wird, die vorgegebene Gewichtskapazität eines Rucksacks dabei aber nicht überschritten wird. Beim Multiplen Rucksackproblem existiert nicht nur ein einzelner Rucksack sondern es sind mehrere vorhanden. Ziel ist es nun, die Gegenstände unter Berücksichtigung der jeweiligen Kapazitätsgrenzen auf die Rucksäcke zu verteilen und dabei wieder maximalen Nutzen zu erreichen.

[4] Das Problem tritt so auch in anderen Bereichen auf. Ein häufiger Anwendungsfall ist zum Beispiel die Erstellung von Pfaden für Bohrer beim Herstellen komplexer Schaltkreise.

[5] Ein Hamilton-Kreis ist ein geschlossener Pfad in einem Graph, der jeden Knoten des Graphs genau einmal besucht.

[6] Jedes entsprechende Problem mit unvollständigem Graph lässt sich in eines mit vollständigem Graph überführen, das die selbe optimale Lösung hat. Dazu werden die fehlenden Kanten, mit ausreichend hohem Gewicht versehen, hinzu gefügt.

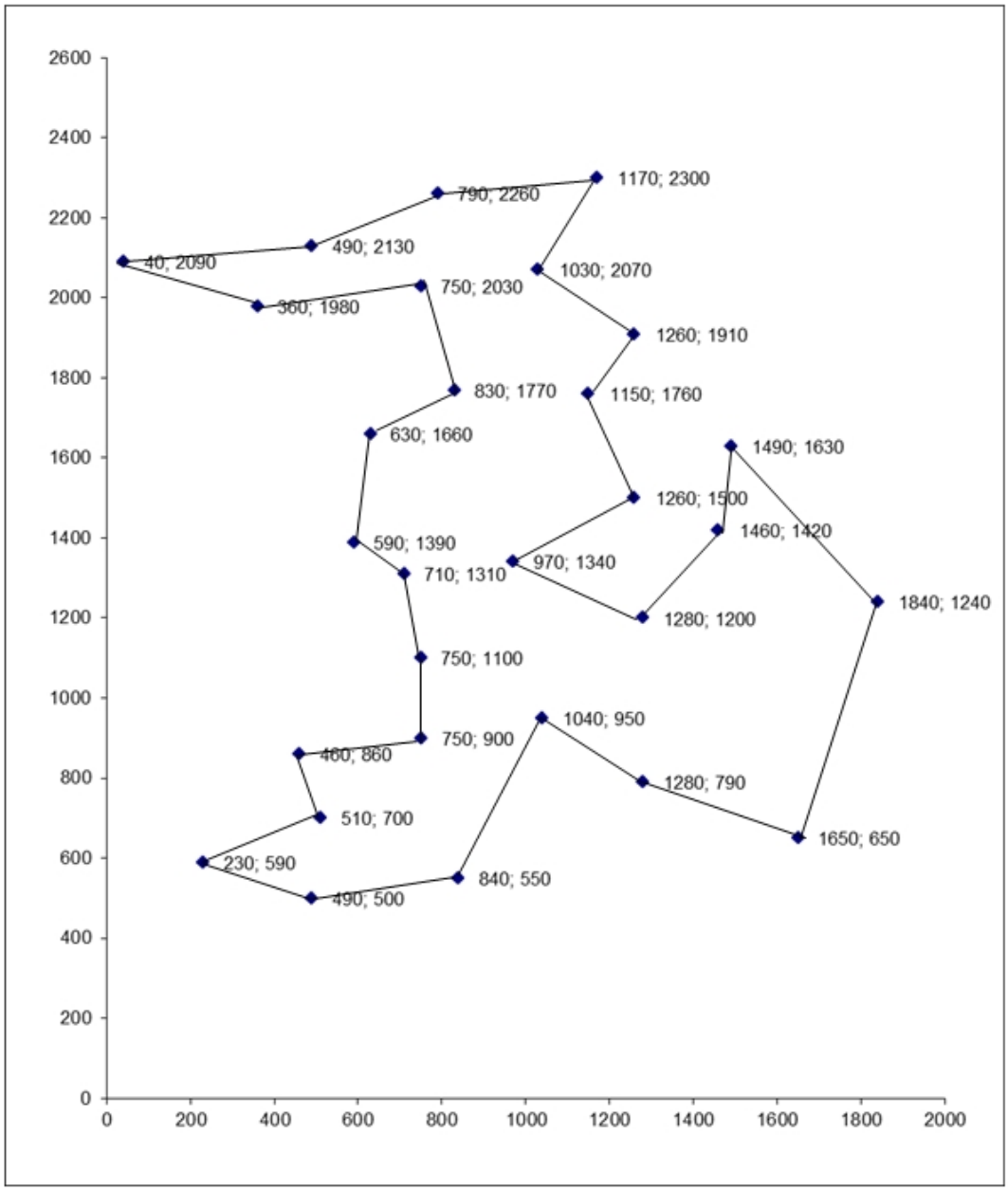

Abbildung 2.3.: TSP bay29 aus der TSPLIB (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/), bestehend aus den 29 größten Städten in Bayern. Bei der eingezeichneten Route handelt es sich um die optimale Lösung.

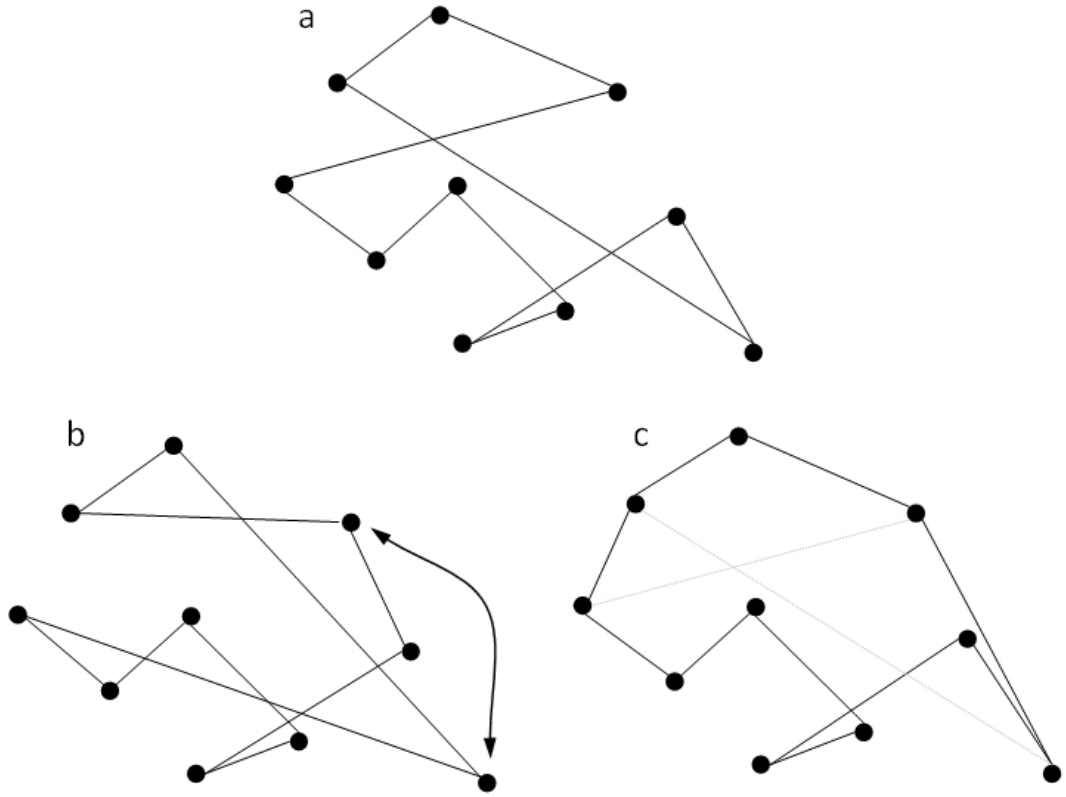

Abbildung 2.4.: Eine mögliche Lösung für ein TSP (**a**), sowie jeweils eine ihrer Nachbarlösungen, einmal erzeugt über den Tausch zweier Städte (**b**), einmal über den Tausch zweier Kanten (**c**).

Während sich bei einem kontinuierlichen Optimierungsproblem die Nachbarschaft einer Lösung x leicht als die Menge aller Punkte y die einen kleineren euklidischen Abstand von x haben als ein zuvor definiertes ϵ, definieren lässt, ist bei den meisten kombinatorischen Problemen eine Nachbarschaftsdefinition nicht so naheliegend. Häufig existieren für das selbe Problem auch mehrere verschiedene mögliche Nachbarschaftsdefinitionen.

Eine mögliche Nachbarschaftsdefinition für das TSP ist beispielsweise, alle die Lösungen als Nachbarn einer Route x zu betrachten, die durch das Vertauschen zweier beliebiger Städte in der Route entstehen (vgl Abbildung 2.4b). Dies bedeutet, eine Lösung eines symmetrischen TSP mit n Städten hat damit $\frac{n(n-1)}{2}$ Nachbarn, die Nachbarschaft ist also sehr groß. Eine andere Möglichkeit der Nachbarschaftsdefinition ergibt sich, wenn statt der Städte zwei Kanten so vertauscht werden, dass die eine Kante i den Anfangspunkt der anderen Kante j als Endpunkt übernimmt, während der alte Endpunkt von i zum neuen Anfangspunkt von j wird (*2-interchange*, vgl. Abbildung 2.4c). Neben diesen beiden sind noch eine Reihe weiterer Nachbarschaftsbeschreibungen denkbar.

Eine gemeinsame Eigenschaft der meisten kombinatorischen Probleme ist die sogenannte *kombinatorische Explosion*. Diese bedeutet, dass die Größe von X mit zunehmender Größe n der Probleminstanz unverhälnismäßig stark anwächst. In den meisten Fällen ist $|X|$ eine Funktion der Fakultät der Problemgröße n. Für das symmetrische TSP mit n Städten beispielsweise hat der Lösungsraum die Größe

$$|X| = \frac{(n-1)!}{2} \tag{2.1}$$

Für $n = 6$ bedeutet dies 60 verschiedene Lösungen, für $n = 7$ schon 360 und für $n = 20$ gibt es $6 * 10^{16}$ verschiedene Routen.

Für kleine Probleminstanzen lässt sich das Optimum eines kombinatorischen Optimierungsproblems also verhältnismäßig leicht durch Ausprobieren aller möglichen Lösungen finden. Dieses Vorgehen bezeichnet man als Vollständige Enummeration. Für größere Probleminstanzen kommt dies aufgrund der Größe des Lösungsraumes allerdings nicht mehr in Frage.

2.3. NP-harte Probleme

Eine weitere Möglichkeit der Unterteilung von Optimierungsproblemen ist nach der Komplexität ihrer Lösbarkeit[7].

Alle Probleme, für deren Lösung sich ein Algorithmus mit polynomieller Zeitkomplexität finden lässt, bilden die Menge P. P ist eine Teilmenge der Menge NP. Diese enthält alle Probleme, die sich mit einer nicht-deterministischen Turing-Maschine (NDTM, vgl. z.B. Garey & Johnson (1979)) mit polynomieller Zeitkomplexität lösen lassen. Vereinfacht ausgedrückt, lässt sich für die Probleme in NP (für ein Entscheidungsproblem) für eine geschätzte Lösung in polynomischer Zeit überprüfen, ob sie das Problem löst, also die Antwort „ja“ liefert.

Nach derzeitigem Wissensstand geht man davon aus, dass P eine echte Teilmenge von NP ist, also $P \neq NP$. Der Beweis für diese Vermutung steht allerdings noch aus.

Unter der Vorraussetzung, dass $P \neq NP$, exixtiert neben P unter anderem noch eine weitere Teilmenge von NP, die der NP-vollständigen Probleme. Als erstes Problem dieser Menge wurde das Erfüllbarkeitsproblem der Aussagenlogik[8] von

[7] Genau genommen gelten die hier aufgeführten Definitionen erst einmal nur für Entscheidungsprobleme, also Probleme deren Antwort „ja“ oder „nein“ ist. Da sich jedes Optimierungsproblem ohne großen Aufwand in ein Entscheidungsproblem überführen lässt (Ersetze „Finde die Lösung x, die $f(x)$ minimiert!“ durch „Gibt es eine Lösung x, für die $f(x) < B$?“), gelten sie dementsprechend auch für Optimierungsprobleme (vgl. Garey & Johnson (1979)).

[8] engl. *Satisfiability Problem*, SAT. Eine Instanz dieses Problems besteht aus einer logischen Verknüpfung C mehrerer boole'scher Variablen u_i, z.B. $C = (u_{17} \vee \bar{u_{37}} \vee u_{73}) \wedge (\bar{u_{11}} \vee \bar{u_{56}}) \wedge \ldots \wedge (u_2 \vee u_{43} \vee \bar{u_{77}} \vee \bar{u_{89}} \vee \bar{u_{97}})$ (Beispiel aus (Michalewicz & Fogel, 2004)). Ziel ist es, eine

Cook (1971) identifiziert. Später wurde für eine Reihe anderer Probleme nachgewiesen, dass sie NP-vollständig sind (für einen Überblick siehe z.B. Garey & Johnson (1979)). Alle NP-vollständigen Probleme lassen sich durch polynomielle Transformationen ineinander überführen. Das heißt, es gibt einen Algorithmus, der mit polinomieller Zeitkomplexität das eine Problem in das andere umwandeln kann. Die Menge aller Probleme, die sich durch polynomielle Transformation in ein NP-vollständiges Problem überführen lassen, egal ob sie Teil der Menge NP sind oder nicht, bezeichnet man als NP-hart.

Gemeinsam ist allen NP-harten Problemen, dass sie nicht mit polynomiellen Algorithmen exakt lösbar sind, sondern nur mit exponentieller Zeitkomplexität. Lösungswege für die exakte Lösung sind dabei meistens nur Abwandlungen der vollständigen Enummeration (Garey & Johnson, 1979), die zum Beispiel durch intelligente Unterteilung des Lösungsraumes bestimmte Lösungen gar nicht untersuchen müssen (z.B. Branch-and-Bound). Sowohl beim TSP als auch beim Rucksackproblem und dem multiplen Rucksackproblem (vgl. Abschnitt 2.2) handelt es sich um NP-harte Probleme.

2.4. Lösungsstrategien für NP-harte Probleme

Für große Probleminstanzen ist eine vollständige Enummeration nicht praktikabel. Andere exakte Verfahren, wie etwa Branch-and-Bound, setzen ein großes Wissen über die Struktur des Lösungsraumes voraus. Ohne dieses Wissen laufen diese Algorithmen wieder auf eine vollständige Enummeration heraus.

Eine Alternative ist es, nicht mehr nach dem wirklichen Optimum zu suchen, sondern zu versuchen, Lösungen zu konstruieren, die diesem Optimum möglichst nahe kommen. Verfahren zur Konstruktion solcher Lösungen nennt man allgemein Heuristiken.

Eine *Metaheuristik* ist eine allgemeine Lösungsstrategie (das heißt, sie kann auf jede Problemstellung angewandt werden), die sich auf intelligente Weise einfacher problemspezifischer Heuristiken bedient und dabei die Suche in vielversprechende Regionen des Lösungsraumes führt (Dorigo & Stützle, 2004).

Die problemspezifische Heuristik ist dabei aufgrund des No-Free-Lunch-Theorems (Wolpert & Macready, 1997) notwendig, das besagt, dass kein Algorithmus über alle Probleme hinweg betrachtet besser sein kann, als eine zufällige Suche. Dem Suchalgorithmus muss also zusätzliches Wissen über das Problem mitgegeben werden.

Bei vielen Lösungsansätzen wird zusätzlich zu der globalen Suchstrategie, die den gesamten Lösungsraum durchsucht, eine lokale Suche hinzugefügt. Die lokale

Belegung der Variablen u_i so zu finden, dass C wahr ist (bzw. als Entscheidungsproblem, herauszufinden, ob eine solche Belegung existiert).

Suche findet nur in der unmittelbaren Nachbarschaft einer bestimmten Lösung statt, die durch ein anderes Verfahren erzeugt wurde (zufällig oder durch einen beliebigen Konstrukrionsalgorithmus). Jeweils durch Suche in der Nachbarschaft wird die Lösung schrittweise verbessert, bis ein lokales Optimum erreicht wurde (oder eine zuvor festgelegte Anzahl an Konstruktionsschritten durchgeführt wurde).

Ein wichtiges Kriterium, das über die Qualität der verschiedenen Lösungsstrategien entscheidet, ist die richtige Balance zwischen der Ausnutzung bereits bekannter Informationen, beispielsweise aus den in vorausgegangenen Iterationsschritten gefundenen Lösungen (*exploitation*), und der freien Erkundung des gesamten Lösungsraumes (*exploration*). Eine zu starke Gewichtung der *exploitation* führt dazu, dass die Suche sich in einem lokalen Optimum verfängt. Dagegen führt eine zu starke Gewichtung der *exploration* dazu, dass eine reine ungerichtete Zufallssuche durchgeführt wird. Nur durch eine ausgewogene Balance zwischen diesen beiden Extremen ist eine gezielte Suche nach dem Optimum möglich (vgl. Abbildung 2.5). Dies ist Voraussetzung dafür, dass ein Suchalgorithmus in effizienter Weise gute Lösungen findet.

Im Rest dieses Abschnitts werden nun einige Heuristiken und Metaheuristiken kurz vorgestellt. Die in dieser Arbeit verwendeten Genetischen Algorithmen und Ameisenalgorithmen werden in den Kapiteln 5 und 6 ausführlich behandelt.

2.4.1. Greedy-Algorithmus

Beim Greedy-Algorithmus[9] handelt es sich wohl um die einfachste heuristische Lösungsstrategie.

Eine Lösung des Problems wird dabei Schritt für Schritt konstruiert (z.B. beim TSP durch schrittweises Hinzufügen einzelner Städte zur Route). Gibt es dabei für einen Schritt mehrere Wahlmöglichkeiten, so wird diejenige gewählt, die lokal „besser“ ist. Für das TSP ist dies beispielsweise diejenige von den bisher noch nicht gewählten Städten, die der zuletzt gewählten Stadt am nächsten liegt.

Dabei verhält sich der Algorithmus sehr „kurzsichtig“: Für die Entscheidungsfindung wird immer nur der direkt nächste Schritt der Lösungskonstruktion herangezogen. Konsequenzen, die eine Entscheidung auf den weiteren Lösungsverlauf haben kann, werden nicht berücksichtigt. Bei einem TSP hat dies beispielweise zur Folge, dass ganze Gruppen von Städten bei der Lösungskonstruktion zunächst ausgelassen werden, die später über unverhältnismäßig weite Strecken noch erreicht werden müssen. Dies passiert dann, wenn die dem momentanen Standpunkt nächstgelegene Stadt näher an der Mehrzahl der noch nicht besuchten Städte liegt als an einer etwas abgelegeneren Gruppe und die Konstruktion damit von letzterer wegführt (siehe Abbildung 2.6).

[9] engl. *greedy*: gierig

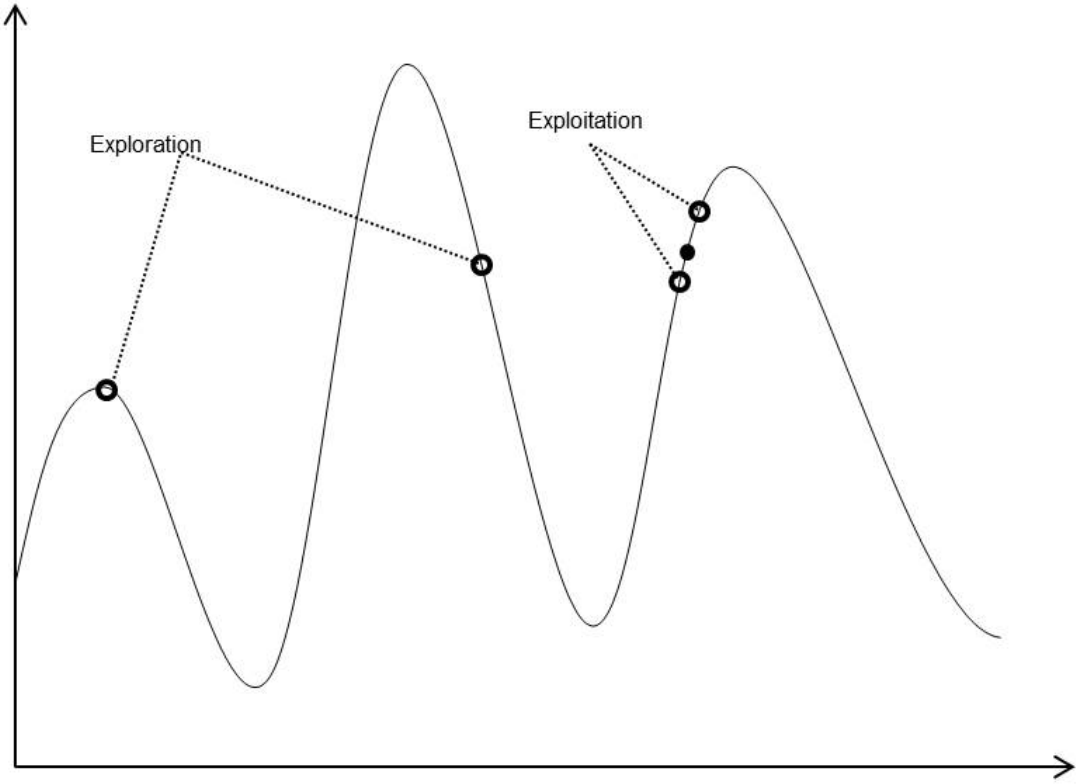

Abbildung 2.5.: *exploration* und *exploitation* auf einer exemplarischen eindimensionalen Fitnessfunktion. Die *exploitation* nutzt die in der durch einen schwarzen Punkt gekennzeichneten bekannten Lösung vorhandenen Informationen. Dadurch wird vornehmlich deren Nachbarschaft abgesucht. Ein rein darauf basierender Optimierungsalgorithmus konvergiert zu dem nächstgelegenen lokalen Optimum. *exploration* bedeutet die freie Erkundung des Lösungsraumes. Damit kann die Suche lokalen Optima entkommen. Eine zu starke *exploration* verhindert allerdings eine gezielte Suche.

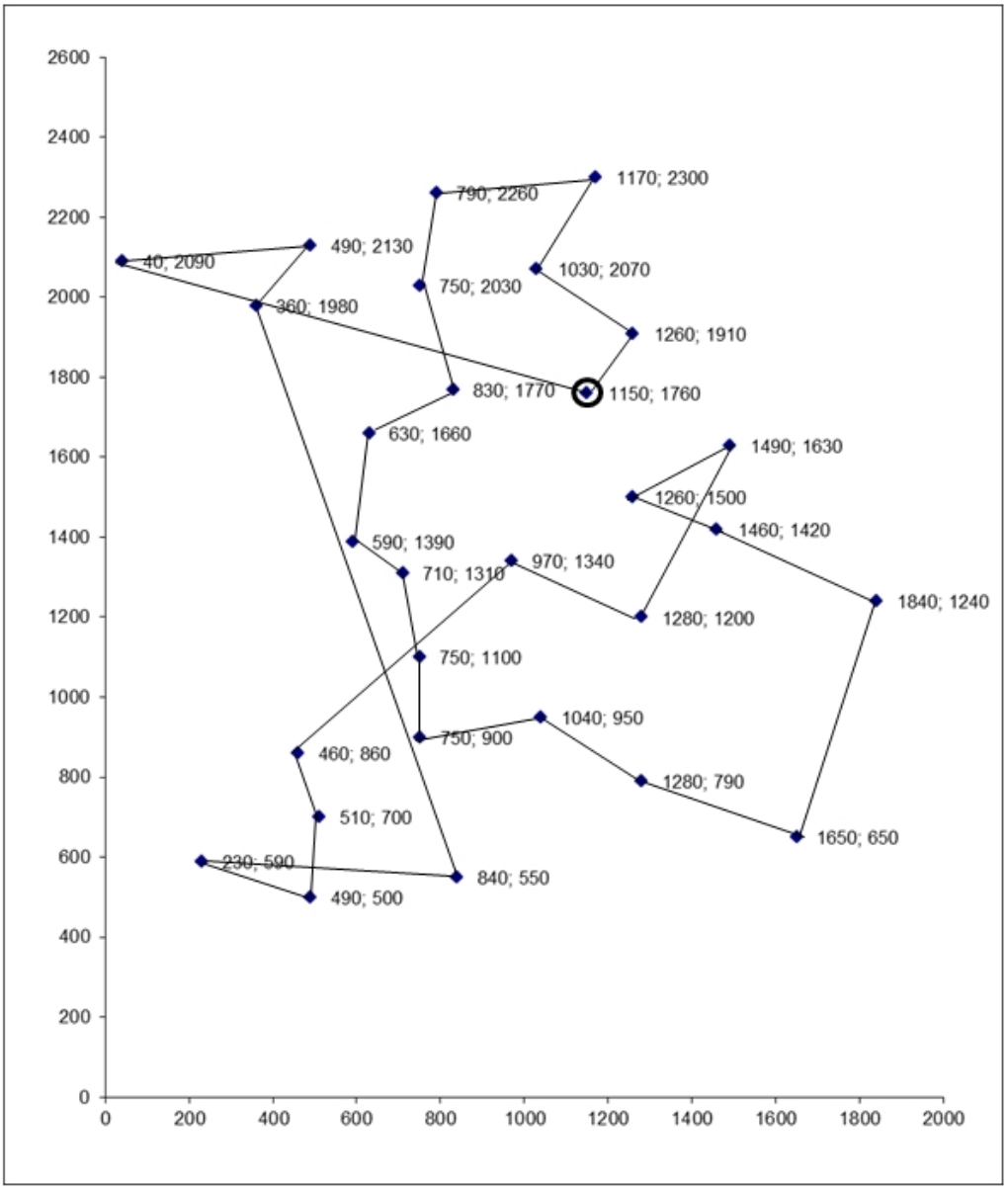

Abbildung 2.6.: Greedy Lösung für das TSP bay29 (vgl. Abbildung 2.3) ausgehend von der markierten Stadt. Es ist deutlich zu erkennen, dass durch kurzsichtige Entscheidungen ganze Gruppen von Städten zunächst ausgelassen werden und später über unverhältnismäßig lange Strecken erreicht werden müssen.

Aus diesem Grund ist der Greedy-Algorithmus auf den meisten Probleminstanzen nicht in der Lage, (lokal oder global) optimale oder auch nur fast optimale Lösungen zu finden. Gleich auf welcher Nachbarschaftsberschreibung gibt es in der Nachbarschaft einer Greedy-Lösung fast immer Lösungen, die besser sind als diese. Je nach gewähltem Startpunkt konstruiert der Greedy-Algorithmus zudem unterschiedliche Lösungen.

Feo & Resende (1989) entwickelten mit GRASP (Greedy Randomized Adaptive Search Procedures[10]) eine Metaheuristik, die auf dem Greedy-Algorithmus aufbaut: Der Algorithmus durchläuft mehrere Iterationen. Jede dieser Iterationen setzt sich zusammen aus einer Konstruktionsphase und einer nachgeschalteten lokalen Suche. In der Konstruktionsphase wird eine Lösung wie im Greedy-Algorithmus Schritt für Schritt, also zum Beispiel beim TSP durch schrittweises Hinzufügen einzelner Städte, erstellt. Allerdings wird dabei nicht immer die lokal beste Wahl getroffen, sondern zufällig einen Konstruktionsschritt aus einer Kandidatenliste der γ besten Wahlmöglichkeiten ausgewählt[11]. Eine auf diese Weise konstruierte Lösung wird dann als Ausgangspunkt einer lokalen Suche verwendet, um so das nächstgelegene lokale Optimum zu erreichen.

Auch wenn GRASP also, im Gegensatz zum reinen Greedy-Algorithmus, fähig ist, lokale Optima zu finden, so zeigt er doch kein Konvergenzverhalten in Richtung des globalem Optimums, da durch die beschränkten Kandidatenlisten nur ein Teil des Suchraumes abgedeckt wird (Mockus *et al.*, 1996).

2.4.2. Bergsteigeralgorithmus

Als Heuristik für eine lokale Suche wird meistens der Bergsteigeralgorithmus verwendet. Anders als beim Greedy-Algorithmus werden die Lösungen nicht einzeln konstruiert. Stattdessen wird hier immer mit vollständigen Lösungen gearbeitet.

Ausgehend von einer zufällig oder durch eine andere Heuristik erstellten Lösung x wird deren Nachbarschaft abgesucht. Ist die beste der benachbarten Lösungen besser als x, so wird die Suche von dieser aus wiederholt. Sobald die Suche ein lokales Optimum $\hat{x}$ erreicht hat, kann keine bessere Lösung in der Nachbarschaft mehr gefunden werden und die Suche wird abgebrochen (vgl. Abbildung 2.7).

Die Metaheuristik der Iterated Local Search (ILS, Martin *et al.* (1991); Lourenço *et al.* (2003)) basiert auf einer mehrmaligen Wiederholung des Bergsteigeralgorithmus. Dabei wird zunächst ausgehend von einer zufällig erstellten Lösung x ein lokales Optimum $\hat{x}$ gesucht. Diese Lösung wird als vorerst beste gefundene Lösung x_{best} gespeichert. Dann wird $\hat{x}$ mit einer Störungsfunktion in eine neue Lösung x' umgewandelt. Ausgehend von x' findet wieder eine lokale Suche statt.

[10] etwa: Adaptive zufallsbasierte Greedy-Suche

[11] Ein ähnlicher Ansatz wurde auch in einigen Implementierungen von Ameisenalgorithmen verwendet (z.B. Gambardella & Dorigo (1996))

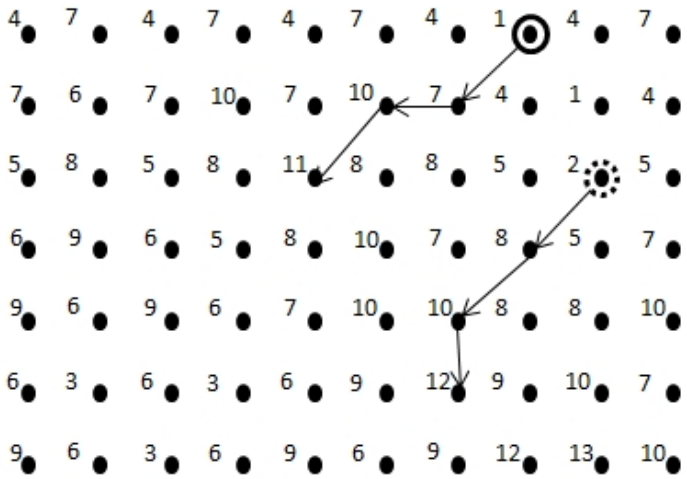

Abbildung 2.7.: Bergsteigeralgorithmus: Dargestellt ist eine Menge von Lösungen eines Problems. Die Nachbarn einer Lösung sind jeweils die waag- und senkrecht sowie diagonal nächstliegenden Punkte. Ausgehend von den beiden markierten Lösungen wird jeweils ein anderes lokales Optimum erreicht.

Ist die dabei gefundene Lösung $\hat{x}$ besser als das bisherige x_{best} so wird dieses durch die neue Lösung ersetzt.

Bei ILS handelt es sich um eine sehr mächtige Metaheuristik, die auf vielen kombinatorischen Optimierungsproblemen zu den besten bekannten Lösungsstrategien zählt (Dorigo & Stützle, 2004). Allerdings ist sie nur bei Problemen mit relativ kleinen Nachbarschaften effektiv.

2.4.3. Tabu-Suche

Ein einzelner Lauf des Bergsteigeralgorithmus findet nur ein lokales Optimum und kann dieses nicht mehr verlassen. Die Suche muss daher mehrmals von verschiedenen Startpunkten aus wiederholt werden. Die Tabu-Suche (Glover, 1986, 1989, 1990) dagegen kann ein einmal gefundenes lokales Optimum wieder verlassen und so innerhalb eines einzigen Laufs mehrere Optima erreichen, darunter mit einer gewissen Wahrscheinlichkeit auch das globale Optimum.

Wie beim Bergsteigeralgorithmus wird ausgehend von einer Startlösung x zunächst eine lokale Suche durchgeführt. Die beste Lösung x' aus der Nachbarschaft von x wird zur Startlösung des nächsten Schrittes verwendet. Anders als beim Bergsteigeralgorithmus kann die Lösung x' dabei auch schlechter sein als x, um dem Algorithmus die Möglichkeit zu geben, ein lokales Optimum zu verlassen.

Damit es dabei nicht zu einem bloßen Hin-und-Her-Springen zwischen zwei Lösungen kommt, bekommt der Algorithmus ein Gedächtnis in Form einer Tabu-Liste.

In dieser wird gespeichert, an welchem Teil der Lösung in den letzten t Schritten Änderungen vorgenommen wurden. Der Teil der Nachbarschaft, der durch eine Änderung an der selben Stelle erreicht wird, ist von der Suche ausgeschlossen, ist also tabu. t muss groß genug gewählt werden, um die Bildung von Zyklen (also die Rückkehr zu einem schon bekannten Optimum) zu verhindern, allerdings klein genug, um der Suche noch genug Freiraum zu lassen.

Eine Problematik bei der Tabu-Suche ist, dass durch die Tabu-Liste bisher nicht besuchte sehr vielversprechende Regionen des Suchraums blockiert sein können. Ein Weg, um dies zu umgehen, ist, wie schon in Glover (1989) vorgeschlagen, die Verwendung von Akzeptanzkriterien (*aspiration criteria*): Eine Lösung, die eigentlich tabu ist, wird als neue Startlösung akzeptiert, wenn sie ein solches Kriterium erfüllt (z.B. besser ist, als alle zuvor gefundenen Lösungen).

Um das Verhalten der Tabu-Suche noch zu verbessern, sind verschiedene zusätzliche Änderungen möglich: So kann die Wahl einer Lösung aus der Nachbarschaft beispielsweise auch probabilistisch abhängig von der Güte der einzelnen Lösungen erfolgen (Glover, 1989). Möglich ist auch die zusätzliche Verwendung eines Langzeitgedächtnisses und das Versehen häufig durchgeführter Änderungen mit einer Strafe (Michalewicz & Fogel, 2004).

2.4.4. Simuliertes Abkühlen

Die Metaheuristik des Simulierten Abkühlens (*Simulated Annealing*, Kirkpatrick *et al.* (1983); Černỳ (1985)) folgt ähnlichen Ansätzen wie die Tabu-Suche. Ihr natürliches Vorbild liegt in dem Verhalten der Moleküle in Flüssigkeiten, z.B. geschmolzenen Metallen, beim Abkühlen: Während sie bei hohen Temperaturen auch energetisch ungünstige Konfigurationen annehmen, geht die Wahrscheinlichkeit dieser Zustände mit sinkender Temperatur zurück. Ein ausreichend langsames Abkühlen vorausgesetzt, das dem System die Möglichkeit gibt, sich immer auf einen Gleichgewichtszustand einzupendeln (vgl. Kirkpatrick (1984)), befinden sich die Moleküle im erstarrten Metall im energetisch günstigsten Zustand.

Auf ein Optimierungsproblem übertragen, besteht dieser energetisch günstigste Zustand im Erreichen des Optimums. Die Molekülbewegungen entsprechen dann lokalen Änderungen der Lösung. Als zusätzliche Größe wird eine Temperatur T eingeführt, die entsprechend eines „Abkühlplanes“ über die Iterationen des Algorithmus langsam verringert wird.

Der eigentliche Suchablauf ist ähnlich dem bei Bergsteigeralgorithmus und Tabu-Suche: Ausgehend von einer Startlösung x wird eine zufällige Nachbarlösung x' erstellt und ausgewertet. Ist x' besser als x so wird es immer als neue Startlösung akzeptiert. Ist x' dagegen schlechter als x, wird es mit einer Wahrscheinlichkeit p akzeptiert, die von den jeweiligen Zielfunktionswerten $f(x)$ und $f(x')$ der beiden Lösungen und der Temperatur T abhängt. Meistens wird dazu die Metropolis-

Verteilung (Metropolis *et al.*, 1953) verwendet (hier für ein Minimierungsproblem):

$$p(x, x', T) = \begin{cases} 1, & \text{wenn} f(x') < f(x) \\ \exp \frac{f(x)-f(x')}{T} & \text{sonst.} \end{cases} \tag{2.2}$$

Simuliertes Abkühlen ist eine recht häufig verwendete Heuristik, die auch auf einige Probleme aus der Ingenieurpraxis angewendet wurde (z.B. Hamm & König (2010); Akbari *et al.* (2012)). Die Hauptschwierigkeit besteht dabei immer darin, einen geeigneten Abkühlplan zu finden: Ein zu schnelles Abkühlen führt zu einer vorzeitigen Konvergenz (das Metall erstarrt in einer Glasstruktur), ein zu langsames Abkühlen bedeutet eine übermäßig lange Rechenzeit.

2.4.5. Populationsbasierte Ansätze

Die in den Abschnitten 2.4.1 bis 2.4.4 beschriebenen Verfahren haben gemeinsam, dass sie jeweils nur eine Lösung gleichzeitig behandeln. Ansätze, die dagegen mit einer Population von Lösungen arbeiten, zeigen gegenüber diesen mehrere Vorteile: Die einzelnen Lösungen können unter einander Informationen austauschen, sei es durch gegenseitige Konkurrenz, sei es durch Zusammenfügen einzelner Elemente verschiedener guter Lösungen. Ein weiterer Vorteil ist die leichte Parallelisierbarkeit, durch Aufteilen in Subpopulationen auf die verschiedenen Prozessoren.

Neben den in dieser Arbeit behandelten Genetischen Algorithmen (Kapitel 5) und Ameisenalgorithmen (Kapitel 6) gibt es noch eine Reihe weiterer populationsbasierter Optimierungsalgorithmen. Beispiele dafür sind Partikelschwarmoptimierung (Kennedy & Eberhart, 1995), der Bienenalgorithmus (Teodorović & Dell´Orco, 2005), der Shuffled Frog Leaping Algorithmus (Eusuff *et al.*, 2006), der Glühwürmchenalgorithmus (Yang, 2009; Gandomi *et al.*, 2011), der Fledermausalgorithmus (Yang, 2010) oder auch der Kuckucksalgorithmus (Gandomi *et al.*, 2012). Allen diesen Ansätzen gemeinsam ist, dass sie eine selbstständige Suche der einzelnen Individuen der Population mit irgendeiner Form von Informationsaustausch verbinden.

2.5. Mehrkriterielle Optimierung

2.5.1. Allgemeines Vorgehen

Anders als bei der einkriteriellen Optimierung gibt es bei mehrkriteriellen Optimierungsproblemen nicht eine sondern mehrere Größen, die optimiert werden sollen. Den Lösungen aus dem n-dimensionalen Suchraum X ist damit also kein einfacher Zielfunktionswert zugeordnet sondern ein Punkt im k-dimensionalen

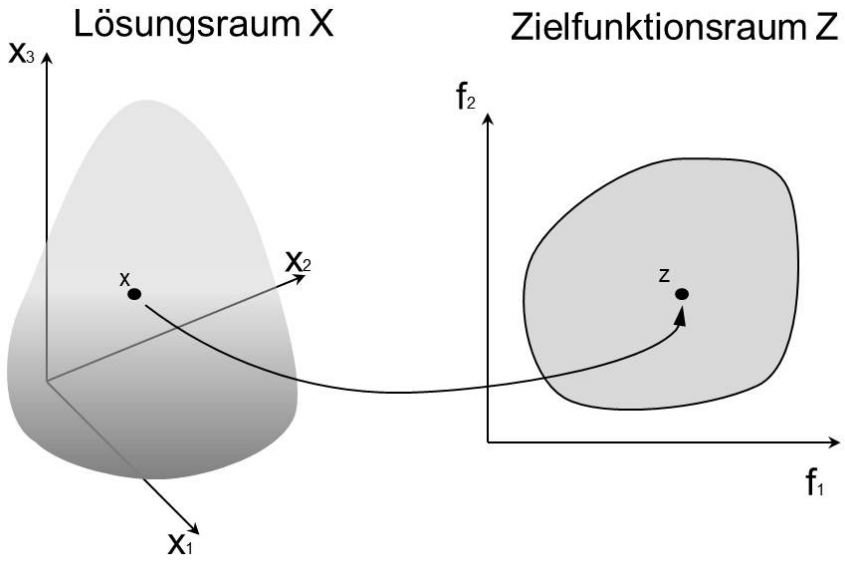

Abbildung 2.8.: Mehrkriterielle Probleme: Den Lösungen x im (hier dreidimensionalen) Lösungsraum X ist jeweils ein Punkt z im (hier zweidimensionalen) Zielfunktionsraum Z zugeordnet (Bild nach Deb (2001)).

Zielfunktionsraum Z (vgl. Abbildung 2.8). Die Zielfunktion $f(x)$ wird also zu einem Vektor $[f_1(x), f_2(x), \ldots, f_k(x)]^T$.

Der ideale Zielfunktionsvektor f^* ist der Vektor im Zielfunktionsraum, dessen Koordinaten den optimalen Werten für die einzelnen Zielkriterien entsprechen. Im Allgemeinen gibt es keine Lösung x^*, die diesem Vektor zugeordnet ist. Nur in dem Fall, dass die verschiedenen Zielkritierien voneinander abhängig sind, ist f^* wirklich zu erreichen, allerdings handelt es sich dann strenggenommen nicht mehr um ein mehrkriterielles Problem.

Da der ideale Zielfunktionsvektor also nicht existiert, handelt es sich bei der mehrkriteriellen Optimierung immer um die Suche nach einer Kompromisslösung, die eine möglichst gute Kombination an Zielfunktionswerten bietet. Welche Kompromisslösung wirklich die beste ist, lässt sich rein mit mathematischen Mitteln nicht bestimmen. Es wird immer ein menschlicher Entscheidungsträger (engl. *decision maker*, DM) benötigt, der festlegt, wie wichtig ihm die einzelnen Zielkriterien sind bzw. welchen Wert er schlechtestens für welches Zielkriterium akzeptiert. Die Wünsche des Entscheidungsträgers können vor (*a priori*), während (interaktiv) oder nach (*a posteriori*) der Optimierung abgefragt werden.

Bei der Entscheidung a priori legt der Entscheidungsträger vor der Optimierung fest, in welcher Reihenfolge die einzelnen Zielkriterien berücksichtigt werden sollen (lexikografische Ordnung) oder welches relative Gewicht den einzelnen Kriterien zugewiesen wird. In beiden Fällen kann die Suche dadurch auf einkriterielle Probleme reduziert werden.

Bei der lexikografischen Ordnung wird das Problem zunächst im Hinblick auf das wichtigste Kriterium hin optimiert. Ausgehend von der so gefundenen Lösung wird versucht, der Reihe nach auch die anderen Kriterien zu verbessern. Ansätze, die so vorgehen finden sich beispielsweise in El Moudani *et al.* (2001) und Barán & Schaerer (2003). Werden vom Entscheidungsträger relative Gewichtungen für die Zielkriterien vorgegeben, so können die verschiedenen Zielkriterien zu einer einzigen Zielfunktion zusammen gefasst werden. Dies kann entweder durch lineare oder auch nicht-lineare Kombination der Zielfunktionen geschehen. Möglich ist auch eine gewichtete Summe der Abweichung der einzelnen Zielfunktionswerte von ihrem jeweiligen idealen Wert (Tchebycheff-Modell).

Ein gemeinsamer Nachteil der a-priori-Methoden ist, dass dem Entscheidungsträger ein großes Wissen über die Problemstruktur abverlangt wird. Auch kann es geschehen, dass eine geringfügig anders gewichtete Lösung, die aber aus nicht ausdrücklich formulierten Gründen attraktiver für den Entscheidungsträger ist als das Optimum der gewählten gewichteten Zielfunktion, so gar nicht gefunden werden kann.

Interaktive Optimierungstechniken erlauben ein Eingreifen des Entscheidungsträgers während des Optimierungsvorganges. Auch hierbei muss dem Entscheidungsträger die Problemstruktur zumindest ansatzweise bekannt sein, sonst kann sein Eingreifen den Suchprozess behindern (Coello Coello *et al.*, 2007). Es kann in Extremfällen auch zu Widersprüchen zwischen den Präferenzen des Entscheidungsträgers kommen.

Bei der Entscheidung a posteriori liefert die Optimierung als Ergebnis eine Menge von guten Kompromisslösungen. Aufgabe des Entscheidungsträgers ist es, aus diesen Lösungen diejenige herauszusuchen, die seinen Wünschen am besten entspricht. Der Vorteil hierbei ist, dass er idealerweise einen guten Überblick über die verfügbaren Lösungen erhält und so leichter entscheiden kann, welchen Kompromiss er am ehesten bereit ist zu tragen.

Aus den genannten Gründen fiel im Rahmen dieser Arbeit die Entscheidung auf die Verwendung von a posteriori Strategien.

2.5.2. Dominanz und Pareto-Optimalität

Als Kriterium zur Bestimmung guter Kompromisslösungen für eine a posteriori Entscheidung des Entscheidungsträgers wird meistens das Dominanzkriterium nach Pareto (1896) herangezogen: Eine Lösung u eines mehrkriteriellen Optimierungsproblem dominiert eine Lösung v genau dann, wenn die Ergebnisse aller Zielfunktionen für u gleich gut oder besser sind als die entsprechenden Werte für v und es mindestens eine Zielfunktion gibt, die für u einen besseren Wert liefert als für v. Man schreibt dann $u \prec v$.

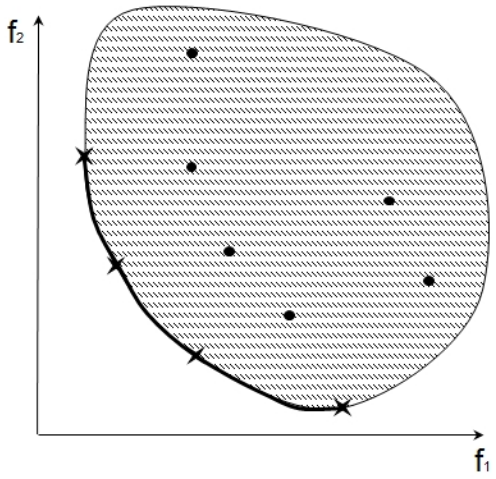

Abbildung 2.9.: Die sternförmig dargestellten Punkte werden bei gleichzeitiger Minimierung der beiden Zielfunktionen f_1 und f_2 in der dargestellten Punktmenge nicht dominiert. Handelt es sich bei der schraffiert dargestellten Menge um die gesamte gültige Zielfunktionsmenge, so liegen die sternförmigen Punkte auf der Paretofront (dickere Linie).

Gibt es in einer Menge S von Lösungen keine Lösung, die die Lösung u dominiert, so gehört u zu der nicht-dominierten Menge in Hinblick auf S. Lösungen, die von keiner Lösung im gesamten Suchraum dominiert werden, bezeichnet man als Pareto-Optima. Die Menge aller Punkte im Zielfunktionsraum, die den paretooptimalen Lösungen zugeordnet sind, bezeichnet man als Paretofront PF (vgl. Abbildung 2.9). Die Lösungen der Paretofront müssen dabei im Suchraum nicht notwendigerweise auch benachbart sein. Es ist vielmehr wahrscheinlich, dass die einzelnen Punkte der Paretofront im Suchraum weit verstreuten Lösungen entsprechen.

Hängen die einzelnen Zielkriterien direkt voneinander ab, z.B. maximale Spannung und maximale Auslenkung eines Kragarms mit veränderlicher Geometrie, dann fällt die Paretofront zu einem einzigen Punkt, der idealen Lösung f^*, zusammen. Ansonsten hängt die Struktur der Paretofront von der Geometrie des gültigen Zielfunktionsraumes ab. Je nach der Form von dessen Oberfläche kann die Paretofront konvex oder konkav sein oder, bei stark konkaver Oberfläche, auch aus mehreren nicht zusammenhängenden Punktmengen bestehen (vgl. Abbildung 2.10).

Die Menge der Pareto-Optima umfasst alle Lösungen, die im Hinblick auf ein Zielkriterium nicht weiter zu optimieren sind, ohne die Lösung für ein anderes Zielkriterium zu verschlechtern. Die Pareto-Optima stellen also die idealen Kompromisslösungen dar. Welche dieser Lösungen gewählt wird, entscheidet sich aus den Wünschen des Entscheidungsträgers.

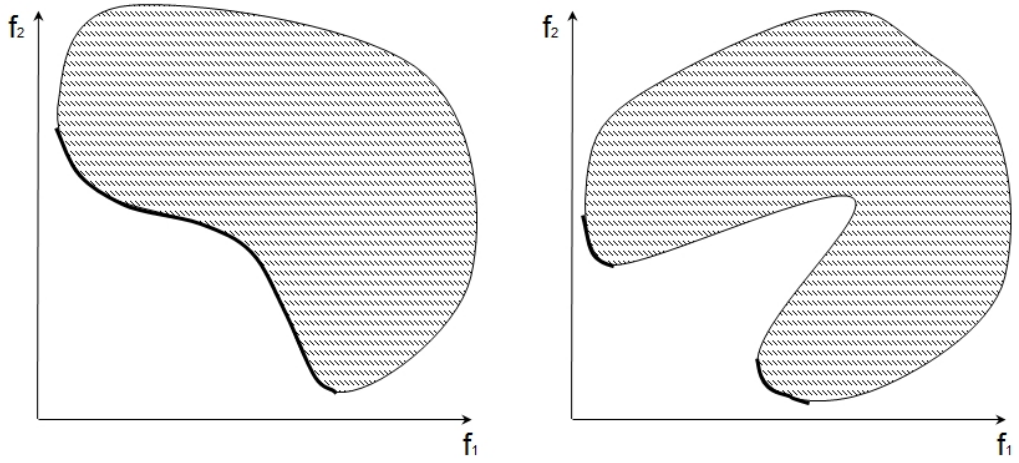

Abbildung 2.10.: Bei gleichzeitiger Minimierung der beiden Zielfunktionen f_1 und f_2 ergibt sich für die links dargestellte Zielfunktionsmenge eine konkave Paretorfront. Für die rechts dargestellte Zielfunktionsmenge besteht die Paretofront aus zwei nicht zusammenhängenden Teilstücken.

Es lässt sich zeigen, dass jede Lösung, die durch Optimierung einer gewichteten Summe der einzelnen Zielfunktionen entsprechend

$$F(x) = \sum_{i=1}^{k} w_i f_i(x) \tag{2.3}$$

gewonnen wird, auch pareto-optimal ist (Miettinen, 1999). Für konvexe Paretofronten gilt außerdem, dass für jede Lösung x der Paretofront ein Wichtungsvektor w existiert, so dass x eine optimale Lösung des entsprechenden Problems nach Gleichung 2.3 ist (Miettinen, 1999). Viele Lösungsansätze für mehrkriterielle Optimierungsprobleme arbeiten basierend auf diesem Prinzip (vgl. auch die Kapitel 5.4.2, 6.4.1 und 6.4.2).

Ziel einer multikriteriellen Optimierung ist es also, Lösungen zu finden, deren Zielfunktionswerte möglichst nah an der Paretofront liegen. Diese sollten dabei möglichst unterschiedlich sein, um die gesamte Paretofront abzudecken. Andererseits sollte die Anzahl dieser Lösungen begrenzt sein, um den Entscheidungsträger nicht zu überfordern.

3. Optimierung im Infrastrukturmanagement

3.1. Konventionelles Vorgehen beim Infrastrukturmanagement

Bauwerke sind kontinuierlich schädlichen Umwelteinflüssen wie Regen, Frost oder Tausalz ausgesetzt, die die Struktur angreifen. Daher sind sie einem Alterungsprozess unterworfen. Im schlimmsten Fall können diese Einflüsse mit der Zeit zu einem Verlust der Gebrauchstauglichkeit oder gar der Standsicherheit führen. Um diesem Risiko entgegenzutreten, müssen vom Bauwerksbetreiber regelmäßig Instandsetzungsarbeiten durchgeführt werden.

In den letzten Jahren, wurde eine Reihe von Programmen entwickelt, die der Verwaltung von zustandsrelevanten Daten von Infrastrukturbauwerken (z.B. Brücken, Unterführungen, Stützwände) auf Bauwerksebene dienen, z.B. SIB-Bauwerke (Abram, 2003), KUBA-MS (Haller & Bascuro, 2006) und BridgeLife (Vesikari, 2008) (siehe auch Abschnitt 4.1.3.2). Neben der Speicherung von Daten können einige dieser Werkzeuge auch Prognosen über die zukünftige Entwicklung des Bauwerkszustandes treffen. Alle diese Programme zur Bauwerksverwaltung sind fähig, Daten zu allen Bauwerken in der Verantwortung eines Betreibers (z.B. einer kommunalen Verwaltung) zu speichern. Davon ausgehend liefern sie jedoch keine weitere Unterstützung bei der Planung, wann eine Instandsetzungsmaßnahme unter Berücksichtigung aller Bauwerke eines Verkehrsnetzes letztendlich am günstigsten anzusetzen ist.

Eine derartige Planung wird heute in der Regel von Hand, basierend auf Erfahrungswerten, durchgeführt (Menke, 2010). Grundlage bilden dabei aus Bauwerksinspektionen (in Ausnahmefällen auch aus Prognosen) gewonnene Zustandsnoten. Aus den Gutachten des Bauwerksprüfers kann auch ein Zeitpunkt bekannt sein, bis zu dem das Bauwerk spätestenfalls instand zu setzen ist.

Neben dem Zustand des Bauwerks gehen in die Überlegungen des Planers noch weitere Punkte ein. So kann jeweils nur eine bestimmte Anzahl an Maßnahmen gleichzeitig durchgeführt werden, da nur eine begrenzte Anzahl an projektverantwortlichen Ingenieuren vorhanden ist.

Ein weiterer wichtiger Punkt, der heute in der Planung von Instandsetzungsmaßnahmen berücksichtig wird, ist die Nutzung von Synergien mit Maßnahmen von anderer Seite, z.B. Gleisinstandsetzungsarbeiten an Straßenbahngleisen, die über das Bauwerk führen. Durch zeitliches Zusammenlegen der Maßnahmen können die Baustelleneinrichtung und eventuell nötige Umleitungserstellungen geteilt und damit Kosten gespart werden. Auch wird dabei vermieden, dass Entschädigungen an die andere Partei aufgrund der entstehenden Behinderungen (z.B. Blockierung einer Straßenbahnroute) zu zahlen sind.

Nur begrenzt finden dagegen die Störungen der Öffentlichkeit, vornehmlich durch Verkehrsbehinderungen, Eingang in die Planung. Sie werden nur insoweit berücksichtigt, dass versucht wird, Maßnahmen an Hauptverkehrsadern so zu legen, dass an der selben Stelle nicht in mehreren aufeinander folgenden Jahren eine Störung entsteht. Zu diesem Zweck werden Maßnahmen in einem Bereich der selben Straße so weit wie möglich zusammen gelegt. Nicht berücksichtigt wird dagegen, ob mögliche Umfahrungen der einen Maßnahme durch eine andere, zeitgleich durchgeführte, Maßnahme blockiert werden und so eine zusätzliche Störung entsteht, die vermeidbar gewesen wäre.

Gar nicht berücksichtigt sind in der heutigen Planung Überlegungen zum Budget. Gerade für öffentliche Verwaltungen ist es aber wünschenswert, die Kosten für Instandsetzungsmaßnahmen über die Jahre zu verstetigen, das heißt, jährlich etwa den selben Betrag für die Instandsetzung von Bestandsbauwerken aufzuwenden um so den Haushalt besser planen zu können und schwierige Verwaltungsmaßnahmen zu vereinfachen.

3.2. Verwandte Arbeiten

Die Frage, wann welches Objekt zu warten bzw. instandzusetzen ist, stellt sich neben der Verwaltung von Infrastrukturbauwerken beispielsweise auch für Öltanks (Li *et al.*, 2002), Produktionsmaschinen (Shum & Gong, 2007) und Kraftwerksgeneratoren (Burke & Smith, 1997; Munõz *et al.*, 1997; Negnevitsky & Kelareva, 1999; Burke & Smith, 2000; Lapa *et al.*, 2000, 2006). Auf diese Probleme wurden bereits erfolgreich Genetische und Memetische[1] Algorithmen angewendet.

Wie bei der Instandsetzung von Straßen und Brücken geht es auch hier darum, einen Instandsetzungszeitplan so zu erstellen, dass die Gebrauchstauglichkeit und Sicherheit der zu wartenden Strukturen erhalten bleibt und gleichzeitig noch der Betrieb aufrecht erhalten wird. Zielkriterium ist dabei im allgemeinen die Minimierung der Kosten, entweder der Reparaturkosten bezogen auf einen begrenzten Zeithorizont (Munõz *et al.*, 1997) oder der Kosten, die über die gesamte Lebensdauer entstehen (Burke & Smith, 1997; Lapa *et al.*, 2006; Shum & Gong, 2007). Andere mögliche Zielfunktionen sind die Minimierung des Versagensrisi-

[1] Genetische Algorithmen in Kombination mit einer lokalen Suche

kos (Munõz *et al.*, 1997) oder die Maximierung der vorhandenen Nettoreserve (Negnevitsky & Kelareva, 1999; Lapa *et al.*, 2000; Li *et al.*, 2002).

Diesen Zielfunktionen gemeinsam ist, dass sie linear sind. Dies stellt einen Gegensatz zu dem in dieser Arbeit behandelten Problem dar, das mit dem Verkehr eine hochgradig nicht-lineare Zielfunktion hat. Außerdem existieren dabei keine nichtlinearen Abhängigkeiten bei gleichzeitiger Instandsetzung zweier Komponenten. Häufig sind die Instandsetzungsintervalle bei diesen Systemen auch so kurz, dass sich zyklische Instandsetzungszeitpläne erstellen lassen.

Ein weiteres Feld der Optimierung von Instandsetzungszeitplänen sind Rohrnetze zur Wasserver- und Abwasserentsorgung. Ziel hierbei ist es, das Netz in gutem Zustand zu halten um den Druckverlust in der Wasserversorgung und das Versagensrisiko der einzelnen Teilstücke zu minimieren. Anders als in Verkehrsnetzen ist hier eine temporäre Totalsperrung von Teilbereichen des Netzes möglich, so dass keine Umleitungen oder ähnliches berücksichtigt werden müssen. Der eigentliche Reparaturvorgang, also die Sperrung, ist damit nicht Bestandteil der Optimierung, die Geometrie des Netzes geht somit nicht in die Planung ein, von Bedeutung ist nur die Zustandsfunktion der Rohre. Dridi *et al.* (2008) und Halfawy *et al.* (2008) wendeten auf diesem Problem zur gleichzeitigen Minimierung der Instandhaltungskosten und Optimierung des Zustands verschiedene mehrkriterielle Genetische Algorithmen an.

Ähnlich wie für Rohrleitungen gestaltet sich das Instandsetzungsmanagement für Eisenbahninfrastruktur. Auch hier ist es möglich, vorübergehend Vollsperrungen einer Strecke vozunehmen (beispielsweise nachts oder an Wochenenden), so dass auch hier die Netzgeometrie zunächst nicht von Bedeutung ist. González *et al.* (2006) optimieren einen solchen Zeitplan auf ein minimales Versagensrisiko der Infrastruktur mit einem auf einem probabilistischen Schädigungsmodell basierenden Entscheidungswerkzeug RCM (Reliability Centered Maintenance). Budai-Balke *et al.* (2009) entwickelten einen Memetischen Algorithmus zur Minimierung der Kosten. Dabei geht durch die Vergabe von Streckenbelegungskosten in die Berechnung der Kosten auch die Störung des Zugverkehrs ein. Dies hat den positiven Effekt, dass Arbeiten am selben Streckenabschnitt möglichst zusammen gelegt werden um Streckenbelegungskosten zu minimieren. Zhang *et al.* (2012) optimieren einen Instandsetzungszeitplan für Eisenbahnstrecken mit Genetischen Algorithmen ebenfalls im Hinblick auf die Kosten. In die Kostenfunktion gehen bei ihnen neben den wirklichen Instandsetzungskosten auch die Kosten resultierend aus einem schlechten Streckenzustand sowie die Reisekosten der Instandsetzungsteams zwischen den verschiedenen Baustellen ein.

Erste Arbeiten zur Instandsetzungsplanung von Straßenbelag betrachten nur eine einzige Straße und versuchen deren Lebensdauerkosten zu minimieren. Tsunokawa & Schofer (1994) lösen dies beispielsweise über dynamische Programmierung.

Auch die meisten Arbeiten, die das gesamte Straßennetz eines Betreibers betrachten, optimieren die Instandsetzungszeitpläne im Hinblick auf die Kosten, teilweise

mit dem Gesamtzustand als zweitem Zielkriterium. Zumindest für kleine Netze ist dies über deterministische Lösungsverfahren möglich. Dabei eingesetzt wurden zum Beispiel ein Markov Decision Model (Guignier & Madanat, 1999), Branch-and-Bound (Ferreira *et al.*, 2002; Ouyang & Madanat, 2004), Integer Programming (Scheinberg & Anastasopoulos, 2010) und ein Parametrischer Algorithmus (zur Bestimmung der Paretofront für Kosten und Zustand, Gao *et al.* (2012)).

Da diese Verfahren bei größeren Probleminstanzen aber schnell ineffektiv werden, wurden schon bald auch Metaheuristiken bei diesem Problem verwendet. Am häufigsten kommen dabei Genetische Algorithmen zum Einsatz (z.B. in Fwa *et al.* (1994); Ferreira *et al.* (2002); Fwa & Farhan (2012)) aber auch andere metaheuristische Verfahren wurden schon verwendet, so benutzen zum Beispiel Wang & Goldschmidt (2008) einen Partikelschwarmalgorithmus, um die Instandsetzung von Straßenbelag im Hinblick auf Kosten und Zustand zu optimieren.

Betrachtet man nur Kosten und Zustand, gestaltet sich das Problem der Instandsetzungsplanung bei Brücken, von anderen Schädigungsfunktionen abgesehen, ähnlich wie beim Straßenbelag. In den meisten Arbeiten zum Thema werden die Instandsetzungszeitpläne für Brücken als mehrdimensionales Optimierungsproblem mit den Zielkriterien Kosten und Zustand bzw. Sicherheit behandelt. Wesentlich weniger Arbeiten als beim Straßenbelag reduzieren das Problem auf eine reine Kostenoptimierung (z.B. Morcous & Lounis (2003, 2005); Elbehairy *et al.* (2006)), wohl aufgrund des größeren Schadenspotentials bei Brücken. Dabei wird häufig auch nicht das gesamte Netz, sondern nur eine einzelne Brücke auf minimale Lebenszykluskosten hin optimiert (Miyamoto *et al.*, 2000; Liu & Frangopol, 2005).

In Holst (2005) wird ein Konzept vorgestellt, dessen Ziel es ist, die Bauwerke im deutschen Fernstraßennetz unter einem begrenzten jahresweisen Budget in einem optimalen Zustand zu halten. Das dort vorgestellte Problem ist dem in dieser Arbeit behandelten sehr ähnlich, allerdings verhält sich der Zustand als Zielfunktion im Gegensatz zum Verkehr linear, was die Lösung vereinfacht. Als Lösungsalgorithmus wurde dort Dynamische Programmierung gewählt.

Morcous & Lounis (2006) verwenden zur Optimierung von Instandsetzungszeitplänen im Hinblick auf Kosten und Zustand Compromise Programming[2]. Die meisten Arbeiten verwenden jedoch Genetische Algorithmen mit einer Binärrepräsentation (vgl. Abschnitt 5.3.1) zur Behandlung dieses Problems. Als mehrkriterieller Genetischer Algorithmus kommen dabei vor allem der NSGA-I (Neves *et al.*, 2006) und NSGA-II (Orcesi & Cremona, 2010; Bocchini & Frangopol, 2011) zum Einsatz. Liu *et al.* (1997) verwenden den Niched Pareto Genetic Algorithm,

[2]Beim Compromise Programming werden verschiedene alternative Problemlösungen erstellt. Für diese werden die Zielfunktionen ausgewertet und dann der Abstand zur idealen Lösung (vgl. Abschnitt 2.5.1) bestimmt. Gewählt wird dann die Lösung mit dem geringsten Abstand. Dieses Verfahren kann auch zur a-postiori Wahl der bevorzugten Lösung aus einer Menge pareto-optimaler Lösungen verwendet werden.

einen Vorläufer des in dieser Arbeit verwendeten Non-Generational Genetic Algorithm for Multi-objective Optimization (vgl. Abschnitt 5.4.1).

Nur wenige Arbeiten zur Instandsetzungsplanung sowohl von Straßen als auch von Brücken berücksichtigen den Einfluss der Instandsetzungsarbeiten auf den Verkehr. In Holst (2005) wird der Verkehr als Nebenbedingung betrachtet: Die möglichen Umleitungen einer Maßnahme dürfen nicht blockiert werden. Da von Holst (2005) nur Fernstraßennetze betrachtet werden, ist diese Bedingung ohne Verkehrssimulator, anders als in feinmaschigen städtischen Netzten, leicht zu überprüfen. Li *et al.* (2011) verwenden einen Ameisenalgorithmus zur Optimierung der Landstraßeninstandsetzung darauf, dass möglichst wenig Verkehr betroffen ist. Dabei wird aber nur die Summe der Belastungen der gesperrten Abschnitte im unbelasteten System betrachtet, dynamische Effekte im Verkehr werden nicht berücksichtigt. Zum genauen Vorgehen bei der Optimierung machen Li *et al.* (2011) leider keine Angaben. Ebenfalls nur die Belastungen im ungestörten System betrachtet Lounis (2005) als drittes Zielkriterium neben Zustand und Kosten bei der Optimierung von Brückeninstandsetzungsplänen. Etwas realistischere Verkehrsmodelle verwenden Orcesi & Cremona (2010) und Bocchini & Frangopol (2011) durch eine eigene Umlegungsfunktion nach dem Wardrop-Modell (Abschnitt 4.2.5.4). Beide Arbeiten betrachten den Verkehr dabei als zweites Zielkriterium neben den Betreiberkosten.

Ng *et al.* (2009) verwenden einen mesoskopischen Verkehrssimulator (RouteSim, basierend auf NETSIM aus Lee (1996)) zur Bestimmung der Reisezeit im durch Instandsetzungsarbeiten (von Brücken und Straßen) gestörten Netz. Diese werden mit den Instandsetzungskosten zu einer gemeinsamen Zielfunktion zusammengeführt, die mit einem binären Genetischen Algorithmus optimiert wird. Dabei wird keine wirkliche mehrkriterielle Optimierung durchgeführt, sondern die verschiedenen Zielfunktionen durch eine a priori Gewichtung miteinander verknüpft (vgl. Abschnitt 2.5). Im Gegensatz zu dem in dieser Arbeit vorgestellten Ansatz wird bei der Terminplanerstellung nicht darauf geachtet, ob einzelne Bauwerke mehrmals in dem erstellten Zeitplan berücksichtigt wurden, was dazu führt, dass im Testszenario einzelne Bauwerke in jedem Jahr des betrachteten Zeithorizonts instand gesetzt wurden.

Lee (2009) koppelt einen Ameisenalgorithmus mit dem mikroskopischen Verkehrssimulator VISSIM (PTV AG, 2011) mit dem Ziel, den Einfluss von Straßenbaustellen auf den Verkehr zu minimieren. Im Gegensatz zu der vorliegenden Arbeit handelt es sich dabei aber um kleine Projekte von wenigen Tagen Dauer, die in einem räumlich eng begrenzten Gebiet stattfinden. Alle betrachteten Maßnahmen müssen im Zeitplan berücksichtigt werden, es werden keine Maßnahmen auf einen Zeitpunkt außerhalb des Betrachtungszeitraumes verschoben. Ziel der Op-

timierung ist dabei hauptsächlich eine optimale Zuweisung von Arbeitergruppen zu den verschiedenen Maßnahmen[3].

Zusammenfasend lässt sich feststellen, dass bisher kein Ansatz zum langfristigen Infrastrukturmanagement in feinmaschigen städtischen Straßennetzen unter realistischer Abbildung des Verkehrs existiert. Zudem werden andere praxisrelevante Fragestellungen, wie die Nutzung von Synergien mit Baustellen von dritter Seite, bisher nicht berücksichtigt.

3.3. Problembeschreibung

In dieser Arbeit liegt das Hauptaugenmerk bei der Optimierung von Instandsetzungszeitplänen auf der Minimierung des Einflusses auf den Straßenverkehr. Neben diesem Zielkriterium können noch weitere Optimierungskriterien in einer mehrkriteriellen Optimierung berücksichtigt werden. Exemplarisch werden hier die optimale Nutzung von Synergien mit Baumaßnahmen von anderer Seite (beispielsweise Straßenbahnbetreiber) und die zeitliche Bündelung von Maßnahmen an Hauptverkehrsadern betrachtet.

Die hier beschriebenen Methoden wurden für das Instandsetzungsmanagement größerer Infrastrukturbauwerke, wie Brücken, Unterführungen und Stützmauern, entwickelt. Geringfügige Modifikationen lassen jedoch auch eine Anwendung bei der Straßenbelagssanierung zu.

Anders als in vielen verwandten Arbeiten werden die Kosten hier nicht als Zielfunktion betrachtet. Stattdessen wird, um den Wunsch nach einem verstetigten Budget zu berücksichtigen, die Randbedingung eingeführt, dass die Ausgaben für Instandsetzungsaufgaben pro Jahr zwischen einer unteren und einer oberen Schranke liegen müssen.

Ebenfalls als Randbedingung berücksichtigt wird der Zustand der Bauwerke. Je nach Beschreibung der Schädigungsfunktion (siehe Abschnitt 4.1.1), lässt sich dies so formulieren, dass für kein Bauwerk eine individuellen Frist zur Instandsetzung überschritten werden darf, oder, dass kein Bauwerk eine schlechtere Zustandsnote als eine festgelegte Schwellenzustandsnote erreichen darf.

Bei der Erstellung der Instandsetzungszeitpläne werden im Rahmen dieser Arbeit zwei vereinfachende Annahmen getroffen:

1. Pro Jahr kann eine genau festgelegte Anzahl an Bauwerken instand gesetzt werden. Der Grund hierfür liegt vor allem in der begrenzten Verfügbarkeit an Bearbeitern: Ein Planungsingenieur in der Verwaltung kann pro

[3]Der gleiche Ansatz wird von Lee (2011) auch auf die Renovierung von Gebäuden unter Aufrechterhaltung der Nutzung angewendet. Ziel hierbei ist, die Personen, die sich zwischen einzelnen Räumen bewegen, möglichst wenig zu stören.

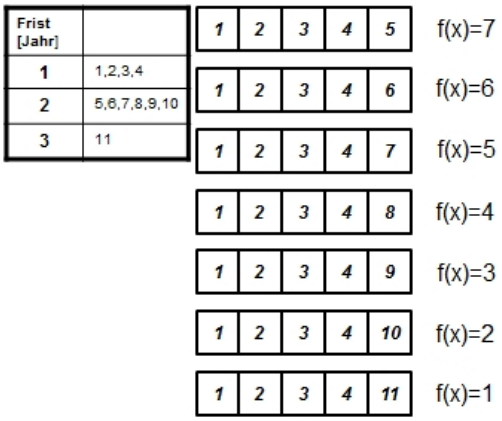

Abbildung 3.1.: In der dargestellten Situation führt eine reine Optimierung für das Jahr 1 dazu, dass im Jahr 2 unter den vorgegebenen Bedingungen kein gültiger Terminplan mehr erstellt werden kann. Eine Minimierung der Zielfunktion für das Jahr 1 führt zur Wahl des Bauwerks Nr. 11 neben den zwingend vorgeschriebenen Bauwerken. Da pro Jahr nur fünf Bauwerke instand gesetzt werden, kann so eines der sechs bis zum Jahr 2 fälligen Bauwerke nicht rechtzeitig instand gesetzt werden. Bei einer Betrachtung beider Jahre würde dies berücksichtigt.

Jahr nur eine feste Zahl an Projekten betreuen, damit ist die Anzahl der in diesem Jahr instandzusetzenden Bauwerke nach oben begrenzt (Menke, 2010). Da das Ziel der Verwaltung andererseits sein sollte, das Gesamtnetz in möglichst gutem Zustand zu halten, sollten auch nicht weniger Bauwerke berücksichtigt werden.

2. Da aufgrund der Frostempfindlichkeit der bei der Instandsetzung verwendeten Baustoffe Instandsetzungsarbeiten in der Regel nur in den Sommermonaten stattfinden, kann man annehmen, dass alle für ein Jahr angesetzten Maßnahmen parallel zur selben Zeit stattfinden.

Es genügt nicht, immer nur die Instandsetzungsarbeiten für das jeweils nächste Jahr zu betrachten und diese für sich allein zu optimieren. Diese Vorgehensweise kann dazu führen, dass es für spätere Jahre keine gültige Lösung mehr gibt. Dies tritt dann auf, wenn die Anzahl der Bauwerke, die spätestens im Jahr j instand gesetzt werden müssen, höher ist als die zulässige Anzahl l an instandzusetzenden Bauwerken pro Jahr, aber der „optimale“ Plan für das Jahr $j-1$ keines dieser Bauwerke berücksichtigt (vgl. Abbildung 3.1). In Holst (2005) wird dieses Szenario als „Sackgasse“ bezeichnet.

Ebenso kann die Optimierung immer nur im Hinblick auf das nächste Jahr dazu führen, dass für spätere Jahre wesentlich schlechtere Lösungen in Kauf genommen werden müssen. Dies tritt beispielsweise dann auf, wenn ein stark belastetes Bauwerk i, das spätestens im Jahr j instand zu setzen ist, als Umleitung für eine andere im Jahr j angesetzte Maßnahme benötigt wird. Wird nun die Instandsetzung

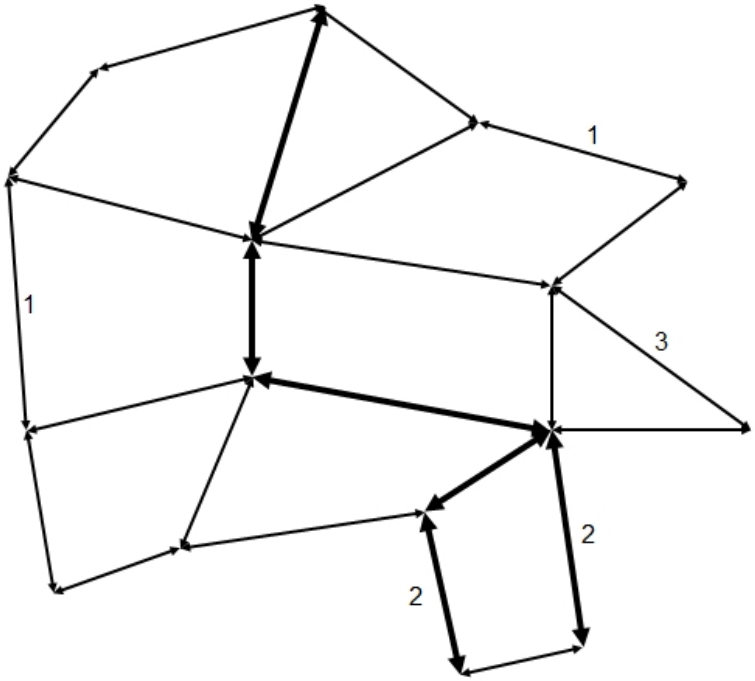

Abbildung 3.2.: In dem dargestellten Netz kann es vorkommen, dass eine Optimierung der einzelnen Jahre des Terminplanes jeweils für sich zu einer über alle Jahre betrachtet schlechteren Lösung führt als eine Optimierung über mehrere Jahre: Angenommen pro Jahr können drei Bauwerke instand gesetzt werden. Da die beiden Bauwerke mit spätesten Instandsetzungszeitpunkt im Jahr 2 sehr stark belastet sind (angedeutet durch die dickeren Pfeile) ergibt sich für das Jahr 1 voraussichtlich eine geringere Störung für den Verkehr, wenn zusätzlich zu den zwingend im Jahr 1 fälligen Bauwerken ein weniger belastetes Bauwerk (z.B. das mit Instandsetzungsfrist im Jahr 3 auf der rechten Seite) gewählt wird. Da die beiden stark belasteten im Jahr 2 fälligen Bauwerke einander gegenseitig als Umleitung dienen, wird ein solcher Terminplan jedoch zu einer sehr starken Störung im Jahr 2 führen.

von i auf das Jahr j gelegt, so ist der Einfluss der Instandsetzungsmaßnahmen auf den Verkehr in diesem Jahr besonders hoch. Setzt man dagegen die Instandsetzung von i im Jahr $j-1$ an, so erhöht sich für dieses Jahr zwar durch das hohe Verkehrsaufkommen an i zwar die Verkehrsstörung, über alle Jahre betrachtet wird der Verkehr aber weniger stark beeinflusst (vgl. Abbildung 3.2).

Aufgrund dieser Überlegungen wird in dieser Arbeit immer ein Terminplan für eine größere Anzahl y von Jahren erstellt. Dies soll aber keinesfalls bedeuten, dass ein einmal erstellter Plan für alle folgenden y Jahre starr bestehen bleibt. Um Notfälle, Schwankungen in der Zahl der verfügbaren Ingenieure und im verfügbaren Budget und ähnliches berücksichtigen zu können, sieht der Ansatz vielmehr vor, dass jedes Jahr aufs Neue ein optimaler Plan über y Jahre erstellt wird, so dass eine Art gleitender optimaler Zeitplan entsteht.

Aus ähnlichen Gründen, wie für die Betrachtung von mehr als einem Jahr, ist es notwendig, zur Erstellung des optimalen Zeitplanes mehr Bauwerke zu betrachten, als letztendlich Teil von diesem Plan sein werden. Die Anzahl n der betrachteten Bauwerke sollte also größer sein als $y * l$. Dies ergibt sich daraus, dass sich manchmal die Berücksichtigung eines Bauwerks, dessen Schädigungsfunktion eine Reparatur nach dem Jahr y zulassen würde, in einem früheren Jahr positiv auf die Zielfunktion(en) auswirken kann[4].

Eine Lösung des Optimierungsproblems besteht damit aus einer Auswahl von jeweils l Bauwerken für jedes von y Jahren aus einer Liste von insgesamt n Bauwerken. Dabei darf jedes Bauwerk höchstens einmal in der Auswahl auftauchen.

Wie bei den meisten kombinatorischen Problemen wächst auch hier die Größe des Lösungsraumes mit zunehmender Problemgröße, hier beschrieben durch die Größen n, y und l, unverhältnismäßig stark an (vgl. Kapitel 2.2). Die Anzahl aller möglichen Lösungen berechnet sich als

$$|X| = \binom{n}{l} * \binom{n-l}{l} * \cdots * \binom{n-(y-1)l}{l} = \prod_{j=0}^{y-1} \binom{n-jl}{l} \tag{3.1}$$

In Abbildung 3.3 ist die Anzahl der möglichen Lösungen für $y = 5$ für verschiedene l abhängig von n dargestellt.

Wie viele dieser Lösungen gültig sind, ist von der jeweiligen Probleminstanz abhängig. Eingang darin findet die Anzahl der pro Jahr spätestens instand zu setzenden Bauwerke[5], die Enge der Budgetgrenzen sowie die Instandsetzungskosten der einzelnen Bauwerke.

Wie im Abschnitt 2.2 erwähnt ist dieses Problem eng mit dem Multiplen Rucksackproblem verwandt (und damit auch NP-hart): Den Rucksäcken entsprechen dabei die Jahre. Auf diese müssen die Bauwerke so verteilt werden, dass der Nutzen maximal (der Einfluss auf den Verkehr minimal) ist. Die Bauwerke haben dabei alle das „Gewicht“ 1 und die Kapazität der einzelnen Jahre entspricht jeweils l. Anders als im klassischen Rucksackproblem ist die Nutzenfunktion jedoch nicht linear. Zudem existieren zusätzlich zu den Rucksackkapazitäten noch weitere Randbedingungen (Bauwerk x muss in „Rucksack“ 1, 2, oder 3, darf aber nicht in „Rucksack“ 4 oder 5). Beides erschwert die Lösung des Problems.

3.3.1. Zielfunktionen

In dieser Arbeit werden drei Zielkriterien betrachtet:

[4] Dies gilt natürlich nur, wenn die Anzahl der bis spätestens zum Jahr y instandzusetzenden Bauwerke kleiner ist als $y * l$. Ist dies nicht der Fall, gibt es allerdings keinen Spielraum für eine Optimierung.

[5] Wenn diese für jedes Jahr gleich l ist, so gibt es nur eine gültige Lösung, ist sie für alle Jahre bis zum Jahr y gleich 0, dann sind, in Abwesenheit anderer Randbedingungen, alle Lösungen gültig.

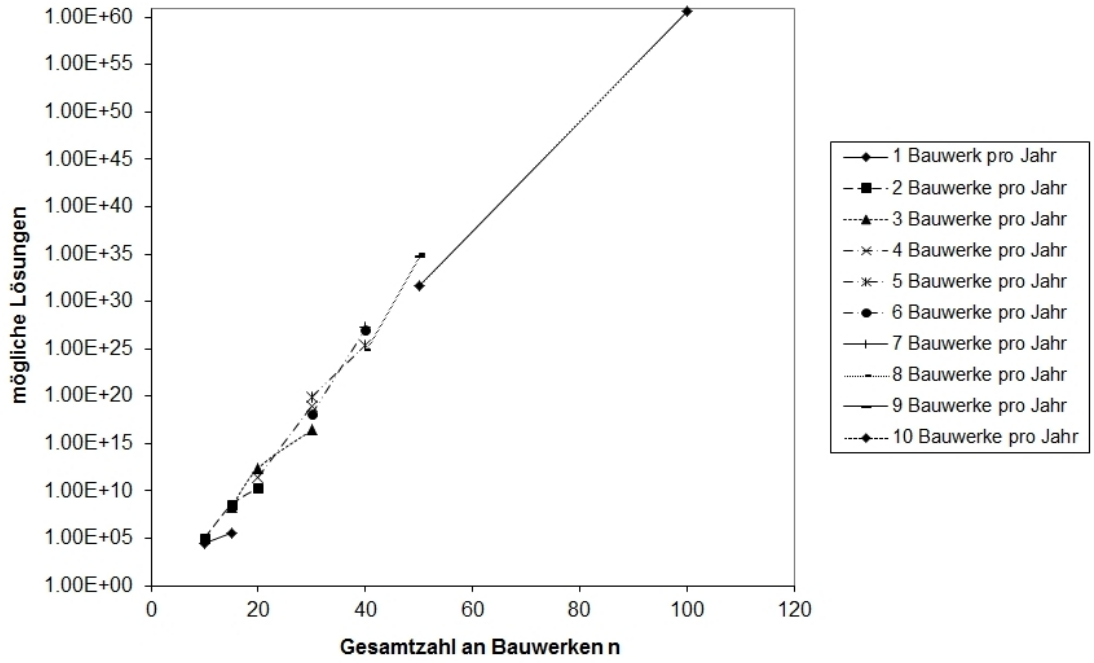

Abbildung 3.3.: Anzahl der möglichen Lösungen für eine Terminplan für $y = 5$ Jahre abhängig von der Menge aller betrachteten Bauwerke n und der pro Jahr wählbaren Bauwerke l.

- Minimaler Einfluss auf den Verkehr
- Optimale Ausnutzung der Synergien mit Maßnahmen von dritter Seite
- Minimale Anzahl an wiederholten Eingriffen an Hauptverkehrsadern

Die Formulierung dieser drei Kriterien als Zielfunktionen wird im folgenden ausgeführt.

Die hier behandelten Kriterien dienen als exemplarische Vorlage zur individuellen Anpassung des hier vorgestellten Konzepts. Neben diesen sind, je nach Schwerpunkt des Bauwerksbetreibers, auch noch weitere Zielkriterien denkbar (z.B. eine Optimierung des durchschnittlichen Zustands aller Bauwerke im Netz).

Im weiteren Verlauf der Arbeit werden zwei verschiedene Problemstellungen behandelt: Ein einkriterielles Problem, bei dem nur der Einfluss auf den Verkehr als Zielkriterium eingeht, sowie ein mehrkriterielles Problem, das alle drei Kriterien betrachtet.

3.3.1.1. Verkehr

Zur Ermittlung des Einflusses eines Instandsetzungszeitplanes auf den Straßenverkehr werden in der hier vorgestellten Arbeit die untersuchten Optimierungswerkzeuge mit einem Verkehrssimulator gekoppelt.

Da die hier verwendeten metaheuristischen Verfahren eine sehr häufige Auswertung (pro Lauf mehrere tausend Mal) der Zielfunktionen verlangen, fiel die Wahl auf einen mesoskopischen Simulator (vgl. Abschnitt 4.2.2). Ein mikroskopischer Ansatz würde zwar eine wesentlich höhere Genauigkeit bei der Abbildung des Verkehrsverhaltens liefern, diese ist aber durch eine erhöhte Rechenzeit erkauft. Der wesentliche Einfluss von baustellenbedingten Störungen im Verkehrsnetz als Optimierungsziel lässt sich auch mit einem mesoskopischen Simulator erfassen.

Verwendet wird hier der kommerzielle Verkehrssimulator VISUM (PTV AG, 2009b). Bei diesem handelt es sich um einen der weltweit meistverwendeten Verkehrssimulatoren. Neben seiner hohen Praxiserprobung und seiner Flexibilität durch die Verfügbarkeit mehrerer verschiedener Verkehrsumlegungsmethoden (siehe auch Abschnitt 4.2.5.4) ist das ausschlaggebende Kriterium für die Verwendung von VISUM in dieser Problemstellung, dass das Programm leicht über eine COM-Schnittstelle fernsteuerbar ist.

Als Maß dafür, wie stark der Einfluss eines Planes zur Instandsetzung auf den Verkehr ist, eignet sich besonders die Anzahl der Fahrzeugstunden [Fzg-h] im gestörten Netz. Dabei handelt es sich um die Summe aller Fahrzeuge multipliziert mit deren jeweiligen Aufenthaltsdauer im Netz. Da sich die Nachfrage im gestörten Netz gegenüber dem ungestörten Netz nicht ändert, handelt es sich dabei also um ein direktes Maß dafür, wie stark die Fahrzeuge im Schnitt durch die Maßnahmen eingeschränkt werden, z.B. durch Staus oder dadurch, dass sie Umwege in Kauf nehmen mussten.

Als Zielkriterium der Instandsetzungsplanoptimierung wurde hier gewählt, dass die Fahrzeugstunden im am stärksten betroffenen Jahr des Zeitplanes minimal sein sollen. Alternativ ließe sich das Problem der Zeitplanoptimierung im Hinblick nur auf den Verkehrseinfluss als mehrkriterielles Problem formulieren, indem die Fahrzeugstunden in allen y Jahren separat minimiert werden.

3.3.1.2. Maßnahmen von dritter Seite

Maßnahmen von dritter Seite sind häufig zeitlich sehr stark fixiert (Menke, 2010). So werden beispielsweise Instandsetzungsmaßnahmen an Straßenbahngleisen regelmäßig alle 15 Jahre durchgeführt. Es lässt sich jedoch davon ausgehen, dass eine gewisse Flexibilität bezüglich dieser Zeiten vorhanden ist, die abnimmt, je weiter die Maßnahme verschoben werden soll.

Als Maß für die Güte eines Zeitplanes x kann hier also gewählt werden, wie weit in ihm die für die von Maßnahmen Dritter betroffenen Bauwerke gesetzten Zeiten von den von dritter Seite vorgesehenen abweichen. Diese Größe über alle Bauwerke aufsummiert soll dann minimiert werden.

3.3.1.3. Hauptverkehrsadern

Zur Behandlung des dritten Zielkriteriums, der möglichst umfassenden Zusammenlegung von Maßnahmen an derselben Hauptverkehrsader, werden Bauwerke, die an diesen wichtigen Straßen liegen, zu Gruppen zusammengefasst.

Um die Qualität eines Zeitplanes bezüglich dieses Kriteriums zu messen, kommen mehrere Möglichkeiten in Frage. Gewählt wurde folgendes Vorgehen: Für jede der zuvor definierten Gruppen von Bauwerken wird zunächst das Jahr im Zeitplan bestimmt, in dem die meisten dieser Bauwerke instand gesetzt werden. Dann wird gezählt, wie viele Bauwerke dieser Gruppe nicht für dieses Jahr eingeplant sind, aber dennoch im Terminplan vorkommen (Bauwerke, die nicht innerhalb der nächsten y Jahre instand gesetzt werden, werden nicht gezählt). Das Zielkriterium ist dann, die Summe dieser Anzahl über alle Bauwerksgruppen zu minimieren.

3.3.2. Randbedingungen

Im Wesentlichen gehen zwei Randbedingungen, die den gültigen Lösungsraum begrenzen, in die Beschreibung des Instandsetzungsproblems ein: Die Sicherheit sowie die Budgetgrenzen. Im Folgenden werden diese näher beschrieben.

3.3.2.1. Sicherheit

Die wichtigste Voraussetzung für den als Optimierungsergebnis ausgegebenen Instandsetzungsplan ist selbstverständlich, dass durch ihn die Sicherheit der Bauwerke gewährleistet ist. Im Klartext heißt dies, dass die im Netz vorhandenen Bauwerke instand gesetzt werden müssen, bevor sie in einen kritischen Zustand geraten, also einsturzgefährdet sind oder ihre Gebrauchstauglichkeit zu stark eingeschränkt ist.

Wie in Abschnitt 3.3 beschrieben, kann das Erreichen des kritischen Zustands entweder direkt über eine für jedes Bauwerk hinterlegte Zustandsfunktion berechnet werden, oder aber diese Zustandsfunktion wurde schon zuvor ausgewertet und für jedes betrachtete Bauwerk existiert eine Frist, bis zu der das Bauwerk spätestens instand zu setzen ist. Im Rahmen dieser Arbeit wurde die zweite Methode gewählt, da sie näher an der derzeitigen Praxis ist: Momentan wird der Bauwerkszustand meistens durch externe Gutachter in Inspektionen bestimmt. Diese

Gutachter geben zumeist auch Empfehlungen dazu ab, bis zu welchem Zeitpunkt das Bauwerk instand zu setzen ist (Menke, 2010).

Zuverlässige Schädigungsfunktionen (vgl. Abschnitt 4.1.1) für die Bauwerke vorausgesetzt, lässt sich das hier beschriebene Verfahren auch direkt mit diesen koppeln. Damit ist auch eine gleichzeitige Betrachtung des gesamten Bauwerksbestands zu jedem Zeitpunkt möglich, während sich mit der herkömmlichen Methode die Planung nur auf einige ausgewählte Bauwerke beschränkt. Sinnvoll wäre dann aber das Einführen einer weiteren Randbedingung, die die Planung einer Instandsetzung für Bauwerke in zu gutem Zustand verhindert.

In den hier beschriebenen Algorithmen wird der Zustand der betrachteten Bauwerke also über das Jahr ihres spätest möglichen Instandsetzungstermines beschrieben. Von Bedeutung sind dabei vor allem diejenigen Bauwerke, für die dieser Termin innerhalb des Betrachtungszeitraumes liegt, also weniger als y Jahre vom Erstellungszeitpunkt des Planes entfernt. Diese Bauwerke werden im weiteren Verlauf als kritische Bauwerke bezeichnet.

Wird mindestens eines dieser kritischen Bauwerke nicht in einem Jahr vor Ablauf seiner individuellen Frist in den Zeitplan aufgenommen, so wird die Sicherheitsrandbedingung verletzt und dieser Zeitplan damit ungültig. Der Grad der Randbedingungsverletzung (und damit die Entfernung der ungültigen Lösung vom gültigen Bereich des Lösungsraumes) lässt sich quantifizieren durch die Anzahl der kritischen Bauwerke, die keinen Termin innerhalb ihrer Frist erhalten haben.

3.3.2.2. Kosten

Wie bereits in Abschnitt 3.3 beschrieben, werden die Kosten hier nicht als Zielfunktion sondern als weitere Randbedingung betrachtet. Ziel dieses Vorgehens ist es, die Geldmenge, die jedes Jahr für Instandsetzungsarbeiten aufgewendet wird, möglichst konstant zu halten, indem nur solche Lösungen zugelassen werden, bei denen die jährlichen Instandsetzungskosten zwischen einer zuvor bestimmten unteren und oberen Schranke liegen[6].

Nach dieser Randbedingung ist eine Lösung dann ungültig, wenn für mindestens eines der y betrachteten Jahre die Budgetgrenzen über- oder unterschritten werden. Der Grad der Verletzung für diese Randbedingung bemisst sich aus der Anzahl der Jahre, für die die Kosten nicht innerhalb des vorgegebenen Intervalls liegen.

In dieser Arbeit wird davon ausgegangen, dass die Instandsetzungskosten eines Bauwerks nur von dessen Größe abhängen. Diese Annahme ist dann zulässig, wenn nur diejenigen Bauwerke betrachtet werden, die bereits in einem schlechten

[6] Es ist hier allerdings darauf zu achten, dass die Intervallgrenzen nicht zu eng gewählt werden, da der gültige Lösungsraum sonst zu stark eingeengt wird und keine gültigen Lösungen mehr gefunden werden können.

Zustand sind, also keine präventiven Maßnahmen durchgeführt werden sollen. Wird dagegen, wie in Abschnitt 3.3.2.1 angesprochen, der gesamte Bauwerksbestand eines Betreibers in die Betrachtung einbezogen, so kann den einzelnen Bauwerken eine vom Bauwerkszustand abhängige Kostenfunktion zugeordnet werden. Derartige Kostenfunktionen werden heute bei der Lebenszykluskostenminimierung einzelner Bauwerke eingesetzt (Miyamoto *et al.*, 2000; Liu & Frangopol, 2005). Eine Anwendung von zustandsabhängigen Kosten auf Netzwerkebene findet sich in Holst (2005).

3.3.3. Zusammenfassung

Zusammenfassend lässt sich das in dieser Arbeit behandelte Problem wie folgt beschreiben:

Gesucht wird ein Terminplan zur Instandsetzung mehrerer durch ein Straßennetz verbundener Bauwerke. Zielkriterien für den optimalen Terminplan lassen sich verschiedene formulieren. Im Rahmen dieser Arbeit wird ein einkriterielles Problem, bei dem der minimale Einfluss auf den Verkehr das einzige Optimierungskriterium ist, sowie ein mehrkriterielles Problem betrachtet. In das mehrkriterielle Problem gehen zusätzliche praxisrelevante Kriterien ein, nämlich die Abstimmung mit Arbeiten Dritter und die gesonderte Behandlung von Hauptverkehrsadern, die möglichst selten Störungen unterworfen werden sollen.

Zur Problembehandlung können zwei vereinfachende Annahmen getroffen werden: Zum einen kann eine feste Anzahl an pro Jahr instandzusetzenden Bauwerken angenommen werden. Außerdem kann vereinfachend davon ausgegangen werden, dass sämtliche für ein Jahr angesetzten Arbeiten zeitgleich stattfinden.

Nicht alle möglichen Zeitpläne stellen gültige Lösungen da. So ist eine Lösung dann ungültig, wenn ein Bauwerk nicht rechtzeitig vor seiner individuellen Instandsetzungsfrist instand gesetzt wird. Zudem lässt sich das Budget als Randbedingung formulieren, so dass für eine Lösung die pro Jahr aufgewendeten Kosten innerhalb vorgegebener Grenzen zu liegen haben.

4. Eingangsgrößen

In diesem Kapitel werden die Eingangsgrößen für die Optimierung beschrieben. Der erste Teil beschäftigt sich mit dem Lebensdauermanagement von einzelnen Bauwerken mit einem Schwerpunkt auf Brücken. Dies liefert die Eingangsgrößen für die Sicherheitsrandbedingung. Der zweite Teil befasst sich mit der Modellierung und Simulation des Straßenverkehrs. Mit einer Verkehrssimulation wird eine Zielfunktionen des in dieser Arbeit behandelten Optimierungsproblems (bzw. die einzige Zielfunktion für das einkriterielle Problem) ausgewertet.

4.1. Lebensdauermanagement auf Bauwerksebene

Das Lebensdauermanagement auf Bauwerksebene liefert die Eingangsgrößen für die Sicherheitsrandbedingung des hier beschriebenen Optimierungsproblems: Die betrachteten Bauwerke müssen instand gesetzt werden, bevor sie in einen Zustand geraten, der ein Gefährdung der öffentlichen Sicherheit darstellt.

In diesem Abschnitt wird dargestellt, wie derartige Gefährdungen prognostiziert werden können. Dabei werden das konventionelle Vorgehen in Deutschland sowie moderne Ansätze beschrieben. Den Abschluss bildet eine Beschreibung des im Rahmen des Forschungsprojekts „Nachhaltig Bauen mit Beton“ (Schießl *et al.*, 2011) an der TU München entwickelten Prototyps für ein prädiktives Lebensdauermanagementsystem (PLMS, Borrmann *et al.* (2012)).

4.1.1. Schäden an Bauwerken

Die RI-EBW-PRÜF (2007) unterscheidet für Ingenieurbauwerke (Brücken, Tunnel, Stützwände,...) zwischen Mängeln und Schäden. Ein Mangel beschreibt eine „Abweichung der Bauwerks- oder Bauteilausbildung vom planmäßigen Sollzustand oder von den zum Prüfzeitpunkt geltenden Regelwerken“. Die Definition eines Schadens beinhaltet dagegen eine zeitliche Komponente: Bei einem Schaden nach RI-EBW-PRÜF (2007) handelt es sich um eine „Veränderung des Bauwerks- oder Bauteilzustandes“. Sowohl Mängel als auch Schäden können negative Auswirkungen auf die Standsicherheit, die Verkehrssicherheit und die Dauerhaftigkeit haben.

Ein Schaden kann aus einem Mangel, einer Überbeanspruchung (z.B. einem Unfall) oder einer Kombination aus beidem entstehen (Schnetgöke, 2008). Typische Schäden an Stahlbetonbrücken[1] sind zum Beispiel (Zilch *et al.*, 2011):

- Carbonatisierung des Betons,
- Chlorideindringen in den Beton,
- Bewehrungskorrosion,
- Spannungsrisskorrosion,
- Ermüdung,
- Frost-Tausalz-Angriff,
- Alkali-Kieselsäure-Reaktion,
- Defizite der Konstruktion,
- Schäden an Konstruktionsteilen und Brückenausstattung.

Laut Schießl & Mayer (2007) machen bei Massivbrücken im deutschen Autobahnnetz chlorid- und carbonatisierungsinduzierte Korrosion über 70% der auftretenden Schäden aus (vgl. Abbildung 4.1). Aus diesem Grund sind dies wohl auch die Schädigungsmechanismen, die am besten untersucht sind.

Nach DIN1045 (2008) werden die Umgebungsbedingungen für Massivbauwerke über Expositionsklassen beschrieben. Dabei werden sechs verschiedene Gefährdungsarten mit jeweils bis zu vier Klassen unterschieden. Die Norm stellt entsprechend dieser Klassen Anforderungen an die Betonqualität und Höhe der Betondeckung um das Schädigungspotential zu minimieren. Die Expositionsklassen können jedoch auch als Eingangsdaten für Schädigungsmodelle verwendet werden. Entsprechende Tabellen zur Bestimmung der Parameter für das Schädigungsmodell von Gehlen (2000) finden sich beispielsweise in Gehlen *et al.* (2008).

Bei reinen Stahlbrücken sind vor allem die Schädigungsmechanismen Korrossion und Ermüdung von Bedeutung. Schädigungsmodelle hierzu finden sich beispielsweise in Kihira *et al.* (2005) und He *et al.* (2012) für Korrosion und in Maier *et al.* (2012) für Ermüdung.

[1] Da Beton- und Spannbetonbrücken den größten Teil (nach Fläche) der Brückenbauwerke im Bestand ausmachen (mehr als 85% der Brücken im deutschen Fernstraßennetz laut Zilch *et al.* (2011)), konzentriert sich die Forschung zu Schadensmodellen vornehmlich auf Massivbrücken.

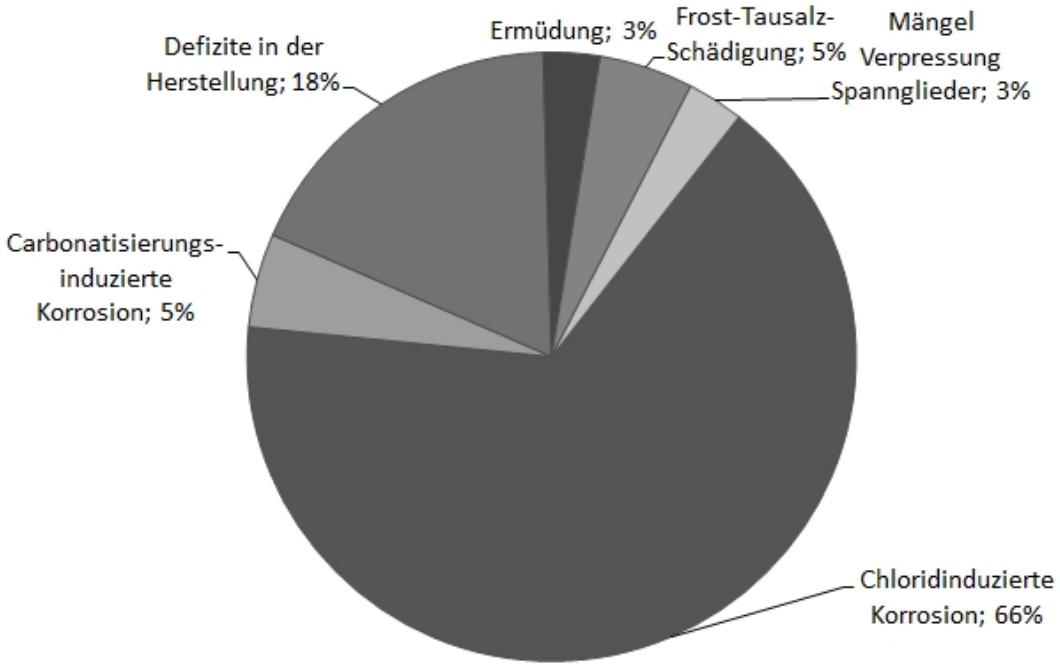

Abbildung 4.1.: Schadensursachen an Betonoberflächen der Brückenbauwerke an deutschen Autobahnen nach Schießl & Mayer (2007).

4.1.2. Bauwerksmanagement nach DIN 1076

4.1.2.1. Bauwerksüberwachung und -prüfung

Die derzeit gültige DIN1076 (1999) sieht regelmäßige Prüfungen an Ingenieurbauwerken vor. Diese gliedern sich dabei in Hauptprüfungen, Einfache Prüfung sowie Prüfungen aus besonderem Anlass.

Eine Hauptprüfung findet zum ersten Mal vor der Abnahme, dann zum Ablauf der Gewährleistungsfrist (nach fünf Jahren) und später regelmäßig alle sechs Jahre statt. Das Bauwerk wird dabei „handnah“ (d.h. z.B. durch Abklopfen) auf Mängel und Schäden (Abschnitt 4.1.1) untersucht.

Jeweils drei Jahre nach einer Hauptprüfung findet eine Einfache Prüfung statt. Diese erfolgt, soweit vertretbar, ohne Besichtigungsgeräte. Schäden und Mängel, die bei der vorausgegangenen Hauptprüfung entsprechend gekennzeichnet wurden, sind besonders zu berücksichtigen. Werden bei einer Einfachen Prüfung erhebliche Schäden festgestellt, so wird sie auf den Umfang einer Hauptprüfung erweitert.

Prüfungen aus besonderem Anlass werden nach besonderen Ereignissen, die Einfluss auf den Bauwerkszustand haben können (z.B. Hochwasser, Verkehrs-

unfälle,...), angeordnet. Vom Umfang her entsprechen diese Prüfungen einer Hauptprüfung.

Prüfungen, die über bloße visuelle und handnahe Methoden hinaus gehen und zerstörungsfreie und zerstörungsarme (z.B. Bohrmehlentnahme) Prüfverfahren einschließen, sind in der aktuellen Norm nur für die „instandsetzungsbegleitende Bauwerksuntersuchung“ im Vorfeld von geplanten Instandsetzungsmaßnahmen vorgesehen.

Neben diesen Prüfungen sieht die DIN1076 (1999) auch jährliche Besichtigungen, sowie etwa vierteljährliche Laufende Beobachtungen vor. Diese erfassen nur offensichtliche Schäden und dienen lediglich der Gewährleistung von Verkehrssicherheit und Standsicherheit.

4.1.2.2. Zustandsbewertung

Die Ergebnisse der Bauwerksprüfungen nach DIN1076 (1999) werden nach RI-EBW-PRÜF (2007) bewertet. Die Bauwerke bekommen danach eine Zustandsnote zwischen 1.0 und 4.0. Dabei bezeichnet eine Note zwischen 1.0 und 1.4 ein Bauwerk in sehr gutem Zustand und eine Note zwischen 3.5 und 4.0 eines in ungenügendem Zustand, dessen Standsicherheit und/oder Verkehrssicherheit nicht mehr gegeben sind (siehe Tabelle 4.1).

Die Zustandsnote des Bauwerks setzt sich aus drei Teilnoten, einer zur Standsicherheit, einer zur Verkehrssicherheit und einer zur Dauerhaftigkeit, zusammen. Diese Noten liegen jeweils zwischen 0 (keine Beeinträchtigung) und 4 (Stand-, Verkehrssicherheit bzw. Dauerhaftigkeit nicht mehr gegeben). Zur Bestimmung der Noten liefert die RI-EBW-PRÜF (2007) einen Katalog an Schadensbeispielen mit deren jeweiliger Bewertung.

Über den Bewertungsschlüssel aus Haardt (1999) wird aus den drei Einzelnoten eine Gesamtnote für jede Bauteilgruppe (Überbau, Unterbau, Vorspannung,...) berechnet. Dabei können unterschiedliche Schadensumfänge über Zu- und Abschläge berücksichtigt werden. Die Note des Bauwerks ergibt sich aus den Zustandsnoten der einzelnen Bauteilgruppen, wieder über Zu- und Abschläge bezüglich des jeweiligen Schadensumfangs.

Den Zustandsnoten ist in der RI-EBW-PRÜF (2007) jeweils ein Handlungsbedarf, auch bezüglich Instandsetzungen, zugeordnet. Genaue Aussagen zum spätesten Zeitpunkt einer Maßnahme werden jedoch nicht getroffen, die Kategorien umfassen „mittelfristige“, „kurzfristige“ und „umgehende“ Instandsetzung. Um aus dem Bauwerkszustand eine Instandsetzungsfrist abzuleiten, wie sie für die Instandsetzungsplanung im Sinne dieser Arbeit erforderlich ist (vgl. Abschnitt 3.3.2.1), ist Ingenieurwissen auf der Seite des Bauwerksprüfers erforderlich.

Note	*Beschreibung*	*Erläuterung*
1.0-1.4	sehr guter Zustand	Standsicherheit, Verkehrssicherheit und Dauerhaftigkeit sind gegeben.
1.5-1.9	guter Zustand	Standsicherheit und Verkehrssicherheit sind gegeben. Die Dauerhaftigkeit mindestens einer Bauteilgruppe kann beeinträchtigt sein. Die Dauerhaftigkeit des Bauwerks kann langfristig geringfügig beeinträchtigt werden.
2.0-2.4	befriedigender Zustand	Standsicherheit und Verkehrssicherheit des Bauwerks sind gegeben. Die Standsicherheit und/oder Dauerhaftigkeit mindestens einer Bauteilgruppe kann beeinträchtigt sein. Die Dauerhaftigkeit des Bauwerks kann langfristig beeinträchtigt werden. Eine Schadensausbreitung oder Folgeschädigung ist langfristig möglich.
2.5-2.9	ausreichender Zustand	Die Standsicherheit des Bauwerks ist gegeben. Die Verkehrssicherheit des Bauwerks kann beeinträchtigt sein. Die Standsicherheit und/oder Dauerhaftigkeit mindestens einer Bauteilgruppe kann beeinträchtigt sein. Die Dauerhaftigkeit des Bauwerks kann beeinträchtigt sein. Eine Schadensausbreitung oder Folgeschädigung ist mittelfristig möglich.
3.0-3.4	nicht ausreichender Zustand	Die Standsicherheit und/oder Verkehrssicherheit des Bauwerks sind beeinträchtigt. Die Dauerhaftigkeit des Bauwerks kann nicht mehr gegeben sein. Eine Schadensausbreitung oder Folgeschädigung ist kurzfristig möglich.
3.5-4.0	ungenügender Zustand	Die Standsicherheit und/oder Verkehrssicherheit des Bauwerks sind erheblich beeinträchtigt oder nicht mehr gegeben. Die Dauerhaftigkeit des Bauwerks kann nicht mehr gegeben sein. Eine Schadensausbreitung oder Folgeschädigung ist kurzfristig möglich. Ein irreperabler Bauwerksverfall kann sich kurzfristig einstellen.

Tabelle 4.1.: Zustandsnoten nach RI-EBW-PRÜF (2007)

4.1.3. Ergänzende Ansätze zum Bauwerksmanagement

Das derzeitige Vorgehen im Lebensdauermanagement nach DIN1076 (1999) weist einige Nachteile auf. So werden Schäden rein visuell oder „handnah“ erfasst, eine Prüfung mit zerstörungsfreien oder zerstörungsarmen Methoden findet nur in Ausnahmefällen statt. Dies führt zu einer stark subjektiven Bewertung des Zustandes. Graybeal *et al.* (2002) zeigten dies für das amerikanische Bewertungssystem, das auf einer zehnstufigen Notenskala aber ansonsten ähnlich arbeitet: Bei der Untersuchung von jeweils den selben sieben Brücken durch 49 erfahrene Prüfingenieure ergaben sich Abweichungen von bis zu 5 Notenstufen, also der halben Skala.

Zudem können Schäden am Bauwerk mit diesem Vorgehen erst dann erfasst werden, wenn sie an der Oberfläche sichtbar sind. Bei vielen Schadensmechanismen beginnt die Schädigung aber schon wesentlich früher (z.B. Bewehrungskorrosion). Durch eine Kombination von zerstörungsfreien Prüfungen und Zustandsprognosen basierend auf Schädigungsmodellen können Schäden schon viel eher entdeckt werden und durch verhältnismäßig kostengünstige Maßnahmen behandelt werden (Schießl & Mayer, 2007).

Verschiedene Forschungsarbeiten aus den letzten Jahren versuchen, diese Mängel zu umgehen. In den folgenden Abschnitten werden einige davon vorgestellt.

4.1.3.1. Lebensdauermangementsysteme

Ein Lebensdauermanagementsystem ist ein computergestütztes System, das einen Bauwerksbetreiber dabei unterstützt, den Zustand eines einzelnen Bauwerks oder des gesamten Bauwerksbestandes über die gesamte Lebensdauer der Bauwerke auf möglichst gutem Niveau zu halten (Schießl & Mayer, 2007). Der Begriff der Lebensdauer im Sinne des Infrastrukturmanagements wird in der Literatur unterschiedlich verwendet: Während einige Arbeiten darunter nur die Phase des Betriebs inklusive Instandsetzungsarbeiten berücksichtigen (Vesikari, 2008) beginnt in anderen die Lebensdauer eines Bauwerks schon mit dessen Planung und endet mit Abriss und Recycling (Schießl & Mayer, 2007; Milachowski, 2008).

Im Sinne dieser Definition ist auch das in dieser Arbeit vorgestellte Optimierungswerkzeug ein Lebensdauermanagementsystem. Da es jedoch nur die Netzwerkebene berücksichtigt, muss es, um wirklich als ein solches eingesetzt werden zu können, mit einem auf Bauwerksebene arbeitenden System gekoppelt werden.

4.1.3.2. Lebensdauermanagementssyteme weltweit

In den letzten Jahren haben viele Länder weltweit eigene Systeme zum Lebensdauermanagement entwickelt. Diese lassen sich in drei Gruppen einteilen. Die

erste Gruppe, zu der zum Beispiel das dänische System DANBRO (Henriksen, 1999) und dessen Weiterentwicklung für Irland Eirspan (Duffy, 2004) sowie der kanadische Forschungscode MMBLMS (Hammad *et al.*, 2006) gehören, besteht aus reinen Werkzeugen zur Datenverwaltung. Ihr Ziel ist es, den Überblick in einem großen Bauwerksbestand zu vereinfachen, so dass der Planer leichter Aussagen über die Dringlichkeit einzelner Maßnahmen treffen kann. Prognosen für die zukünftige Entwicklung können in ihnen nicht gestellt werden.

Die zweite Gruppe an Lebensdauermanagementsystemen besitzt eine eingebaute Prognosefunktion, diese basiert jedoch auf rein deterministischen Schädigungsmodellen. Zu dieser Gruppe gehören beispielsweise die in Kanada entwickelten Systeme BRIDGIT (Hawk, 1999) und SSAM-I (Vanier *et al.*, 2009) oder das von Miyamoto *et al.* (2000) vorgestellte Sytem. Derartige Systeme haben den Nachteil, dass Streuungen der Schädigungsparameter sowohl auf Einwirkungs- als auch auf Widerstandsseite nicht berücksichtigt werden. Daher liefern diese Modelle nur einen gemittelten Zustand für die einzelnen Bauteile, der auch auf der unsicheren Seite liegen kann (Milachowski, 2008).

Für die Lebensdauermanagementsysteme der dritten Gruppe sind probabilistische Prognosefunktionen implementiert. Diese Systeme gliedern sich wiederum in solche, deren Prognose auf Markow-Ketten[2] basiert (z.B. das US-amerikanische PONTIS (Robert *et al.*, 2003), das schweizer KUBA-MS (Haller & Bascuro, 2006) oder das finnische BridgeLife (Vesikari, 2008)), und solche, denen wirkliche probabilistische Schädigungsmodelle zugrunde liegen. Letztere haben bisher noch keinen Eingang in die Praxis gefunden, wurden aber schon in Forschungsprototypen umgesetzt (z.B. Frangopol *et al.* (2001)). Da noch nicht für alle bekannten Schädigungsmechanismen explizite Modelle existieren (Zilch *et al.*, 2011), muss es sich bei Umsetzungen von Systemen des letztgenannten Typs in der Praxis notgedrungen um Mischformen handeln, bei denen nicht vorhandene Modelle durch Markow-Ketten ersetzt werden.

4.1.3.3. Überblick zu Vorschlägen für ein Lebensdauermanagementsystem in Deutschland

In den letzten zehn Jahren hat in Deutschland intensive Forschung zur Entwicklung eines Lebensdauermanagementsystems, das die Nachteile des bisherigen Vorgehens überwindet, stattgefunden. Dabei wurden verschiedene Konzepte erstellt, die im Folgenden kurz erläutert werden. Für einen tieferen Überblick siehe auch

[2]Markow-Ketten sind ein stochastisches Modell zur zeitlichen Entwicklung von Systemen. Sie gehen von der (vereinfachenden) Annahme aus, dass der Zustand eines Systems zum Zeitpunkt t nur von dessen Zustand zum Zeitpunkt $t-1$ abhängt, nicht aber von der weiteren Vergangenheit. Auf Schädigungsmodelle übertragen können Übergangswahrscheinlichkeiten zwischen den verschiedenen Zuständen von einem Zeitpunkt t auf den Zeitpunkt $t+1$ festgelegt und damit aus den jeweiligen Wahrscheinlichkeiten, dass sich ein Bauteil zum Zeitpunkt t in den verschiedenen Zuständen befindet, die Wahrscheinlichkeiten für alle Zustände für alle Zeitpunkte $t+x$ abgeleitet werden.

Zilch *et al.* (2011). Das Konzept von Schießl & Mayer (2007), das die Grundlage des PLMS-Prototypen bildet und damit Ausgangspunkt der hier vorgestellten Arbeit war[3], wird in Abschnitt 4.1.4 ausführlich beschrieben.

Im DFG-Sonderforschungsbereich 477 wurden an verschiedenen Instituten der Technischen Universität Braunschweig Methoden und Strategien zur Sicherstellung der Gebrauchstauglichkeit und Tragsicherheit von Ingenieurbauwerken entwickelt. Ergebnisse dabei waren probabilistische Schadensmodelle und ein Monitoringkonzept zur Bauwerksüberwachung im Hinblick auf diese Schäden (Schnetgöke, 2008) sowie das Progammsystem PROBILAS zur Unterstützung der probabilistischen Modellierung (Klinzmann & Hosser, 2005). Nachteil des Modells ist, dass es sehr stark für jedes einzelne Bauwerk spezialisiert werden muss.

Müller & Vogel (2009) zerlegen ein Brückenbauwerk in einzelne Systemelemente, die dann als Parallel- oder Reihenschaltung zum Gesamtsystem gekoppelt werden, so dass Aussagen zur Auswirkung eines Schadens auf das Gesamtbauwerk getroffen werden können. Jedem Systemelement ist wiederum ein Fehlerbaum zugeordnet, der sich aus einzelnen Schädigungsmechanismen zusammensetzt. Diese Schädigungsmechanismen werden über probabilistische Modelle beschrieben. Ein Nachteil dieses Modells ist, dass die einzelnen Komponenten des Modells keine geometrische Entsprechung am Bauwerk haben, was die Eintragung und Verwertung von Inspektionsdaten erschwert. Zudem muss das Systemmodell für jedes Bauwerk neu formuliert werden.

Im Auftrag der Bundesanstalt für Straßenwesen (BASt) erarbeiteten Zilch *et al.* (2011) ein Konzept, das die herkömmlichen Zustandsnoten der RI-EBW-PRÜF (2007) mit ihrer Aufgliederung in Standsicherheit, Verkehrssicherheit und Dauerhaftigkeit mit probabilistischen Schadensmodellen kombiniert. Dabei machen sie Vorschläge zur probabilistischen Behandlung aller in Abschnitt 4.1.1 aufgeführten Schadensmechanismen.

Mit diesen Schädigungsmodellen wird für das Bauwerk bzw. seine Teile über die gesamte Lebensdauer hinweg jeweils die Wahrscheinlichkeit bestimmt, mit der die einzelnen Zustandsnoten erreicht wurden. Dazu wird das Bauwerk untergliedert in Bauteilgruppen, denen wiederum Schädigungsmechanismen zugeordnet werden. Die probabilistische Berechnung erfolgt auf Ebene der Schadensmechanismen, die gegebenenfalls nach örtlichen Besonderheiten unterteilt werden kann. Für die Jahre, in denen die Schädigungswahrscheinlichkeit einen zuvor festgelegten Grenzwert überschreitet, werden Inspektionstermine angeordnet. Stellt diese Inspektion eine Schädigung fest, so ist das Bauwerk instand zu setzen. Wird kein Schaden gefunden, so wird das Schädigungsmodell entsprechend der Inspektio-

[3]Das in dieser Arbeit vorgestellte Konzept zur Optimierung der Instandsetzungszeitpläne kann ohne weitgehende Änderungen auch mit den anderen hier vorgestellten Konzepten gekoppelt werden.

nergebnisse über ein Bayes'sches Update[4] angepasst, und darüber der nächste Inspektionszeitpunkt bestimmt.

4.1.4. Prototypische Umsetzung eines prädiktiven Lebensdauermanagementsystems

Im Rahmen des Forschungsprojekts „Nachhaltig Bauen mit Beton" (NBB) des Deutschen Ausschusses für Stahlbeton (DAfStb) wurde ein Konzept für ein prädiktives Lebensdauermanagementsystem (PLMS) erstellt und prototypisch umgesetzt (Schießl *et al.*, 2011). Eine der Ideen dabei war, nach dem Prinzip des Building Information Modelling (BIM) die Informationen zum Bauwerksmanagement an ein dreidimensionales Geometriemodell des Bauwerks zu koppeln. Dieses Modell soll die Verwendung gegenüber den aktuell eingesetzten textbasierten Systemen intuitiver machen und helfen Fehlplatzierungen von Daten zu verhindern. In Anlehnung an das in Kanada entwickelte MMBLMS (Hammad *et al.*, 2006) ist auch ein Einsatz auf einem mobilen Gerät direkt bei der Bauwerksinspektion denkbar.

Zentraler Bestandteil des Systems ist eine Datenbank, in der alle Daten, sowohl Grunddaten wie Daten zum Material (z.B. Betongüte, verwendeter Zement,...) und Daten, die erst über die Lebensdauer entstehen (z.B. Messdaten) gespeichert werden. Diese Daten sind unterteilt in Geometriedaten und Metadaten. Alle im System gespeicherten Daten sind dabei mit den Geometriedaten (entweder Volumenkörpern oder Oberflächen) verknüpft und können im Modell lokalisiert werden (Borrmann *et al.*, 2012).

Die Geometrie wird als B-rep-Struktur in Anlehnung an die Datenstruktur des ACIS Geometrie-Kernels (Spatial, 2006) gespeichert. Die Geometrie setzt sich dabei auf unterster Ebene aus Knoten (*Vertices*) zusammen, die durch ihre Koordinaten beschrieben werden. Je zwei Knoten formen eine gerichtete Kante (*Coedge*). Je zwei entgegengerichtete Kanten bilden eine ungerichtete Kante (*Edge*). Eine Folge von ungerichteten Kanten ergibt einen *Loop*. Eine Oberfläche (*Face*) wird von einem oder mehreren Loops umschlossen. Eine Gruppe von Oberflächen formt eine *Shell*, die einen *Lump* umschließen. Der Volumenkörper (*Body*) selbst wird aus einem oder mehreren Lumps zusammengesetzt. Aufgrund der Ebenen Loop, Shell und Lump können so auch Körper mit Löchern abgebildet werden (vgl. Abbildung 4.2). Derartige Modelle können aus Modellen in gängigen CAD-Programmen abgeleitet werden (vgl. Breitenberger (2008)).

Das Bauwerk selber wird entsprechend dem Vorschlag von Schießl & Mayer (2007) in fünf Ebenen untergliedert (vgl. Abbildung 4.3). Die oberste Ebene beschreibt

[4] Bei einem Bayes'schen Update wird die Wahrscheinlichkeit einer Schädigung S_t zum Zeitpunkt t unter der Vorraussetzung, dass zum Zeitpunkt t eine Messung das Ergebnis M_t liefert, über den Satz von Bayes bestimmt zu $p(S_t|M_t) = \frac{p(M_t|S_t)*p(S_t)}{p(M_t)}$ (Straub, 2004).

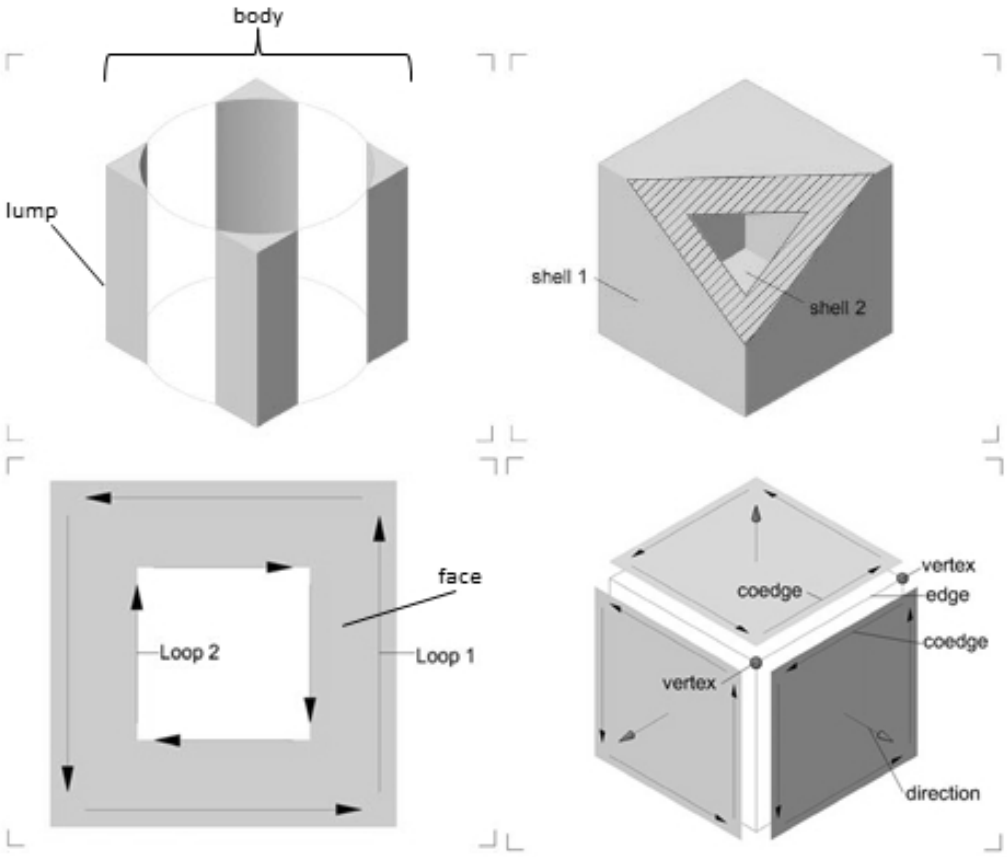

Abbildung 4.2.: Struktur des Geometriemodells (aus Breitenberger (2008)). Links oben dargestellt ist ein Volumenkörper (Body), der aus vier Teilkörpern (Lump) zusammengesetzt ist. Der Körper rechts oben hat zwei Shells. Unten links ist eine Fläche (Face) zu sehen, die von zwei Loops umschlossen wird. Das Bild unten rechts beschreibt anhand einer Würfelecke den Zusammenhang zwischen Edges und Coedges: Durch die Richtung der Coedges wird ein Umlaufsinn um die einzelnen Faces vorgegeben, der bestimmt, welches die Außenseite ist.

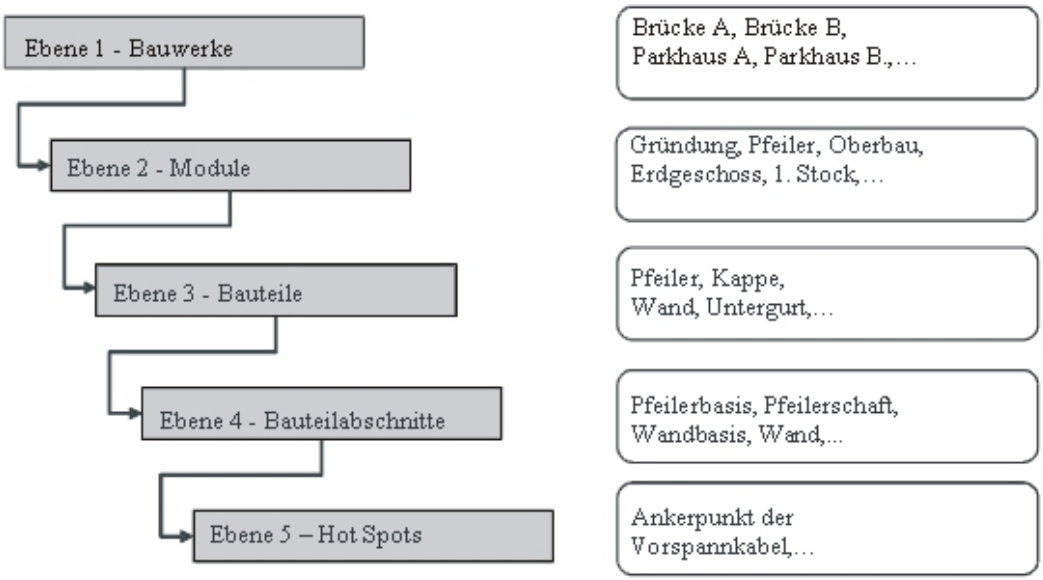

Abbildung 4.3.: Unterteilung eines Bauwerks nach Schießl & Mayer (2007) in fünf Ebenen.

das Bauwerk als ganzes. Dieses wird unterteilt in Module. Ein Modul fasst solche Elemente zusammen, die geometrisch oder aufgrund ihrer Funktion zusammen gehören. Beispiele für Module einer Brücke sind Gründung, Pfeiler, Überbau,...

Die dritte Ebene beschreibt die Bauteile selber. Diese können in den Ebenen 4 und 5 weiter unterteilt werden: Ebene 4 beschreibt Bauteilabschnitte. Diese werden dann eingesetzt, wenn Teile von Bauteilen, die selber keine eigenständigen Bauteile sind, anderen Belastungen ausgesetzt sind als der Rest des Bauteils. Dies gilt zum Beispiel für den Fuß einer Widerlagerwand, der durch Spritzwasser einer erhöhten Tausalzbelastung ausgesetzt ist. Die unterste Ebene wird als „Hot Spots“ bezeichnet. Hierbei handelt es sich um solche Stellen des Bauwerks, die aus konstruktiven Gründen besonders gefährdet sind (z.B. Fugen, Ankerpunkte der Vorspannung) oder an denen bereits eine beginnende Schädigung festgestellt wurde, die bisher aber noch nicht instand gesetzt wurde.

Den Elementen der Ebenen 3 bis 5 kann ein Geometrieelement des Bauwerksmodells zugeordnet werden, so dass sie dort lokalisiert werden können. Dabei handelt es sich um Volumenkörper für die Ebenen 3 und 4, die Elemente der Ebene 5 werden als Oberflächen beschrieben.

Den Volumenkörpern des Bauwerksmodells werden jeweils Materialdaten sowie der Bauzeitpunkt zugewiesen. Die einzelnen Oberflächen der Körper werden, sofern es sich nicht um Schnittflächen handelt, mit Belastungen in Form von Expositionsklassen versehen (Abbildung 4.4). Aus den Materialdaten auf der Widerstandsseite und den Expositionsklassen als Belastung können nun für die einzelnen Oberflächen Prognosen zur Zustandsentwicklung über die Lebensdauer

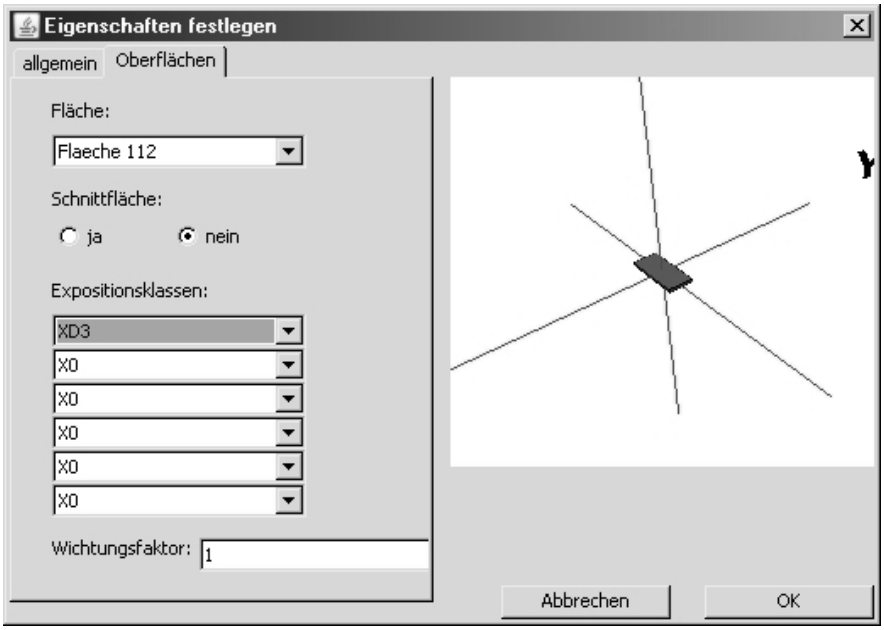

Abbildung 4.4.: Screenshot des PLMS-Prototyps: Der Oberfläche einer durch Tausalz belasteten Betonplatte wird die Expositionsklasse XD3 zugewiesen. Einer Oberfläche können bis zu sechs Expositionsklassen zugewiesen werden. Zusätzlich kann der Oberfläche ein Wichtungsfaktor zugewiesen werden. Am Modell rechts kann überprüft werden, ob die richtige Oberfläche ausgewählt wurde.

erstellt werden (Borrmann *et al.*, 2012). Im Prototyp implementiert sind nur Schädigungsmodelle zur Carbonatisierung und zum Chlorideindringen. Verwendet wurden dabei die Modelle von Gehlen (2000). Andere Schädigungsmodelle lassen sich jedoch entsprechend ergänzen.

Über die Zuweisung von Messdaten aus Inspektionen oder Monitoring zu den Bauteiloberflächen lassen sich neue Erkenntnisse zu deren Zustand gewinnen. So können die unsicheren Modellparameter durch Messungen über die Lebensdauer eingegrenzt werden, so dass die Prognose mit fortschreitender Nutzungsdauer immer genauer wird. Wie im Ansatz von Zilch *et al.* (2011) (vgl. Abschnitt 4.1.3.3) erfolgt auch hier die Einbindung der Messergebnisse über ein Bayes'sches Update (Borrmann *et al.*, 2012).

Die Zustandsprognosen können in Referenz zum 3D-Modell dargestellt werden. Durch eine farbliche Codierung der Zustandsnoten (z.B. grün = sehr guter Zustand bis rot = stark gefährdet) kann dies zu einer intuitiven Erfassung von Schwachstellen des Bauwerks dienen (Abbildung 4.5). In Anlehnung an Schießl & Mayer (2007) wird im Prototyp eine sechsstufige Zustandsnotenskala (1-6, siehe Tabelle 4.2 und Abbildung 4.6) verwendet. Gegenüber der klassischen Skala nach RI-EBW-PRÜF (2007) hat diese den Vorteil, dass mit ihr Schädigungen erfasst werden können, bevor sie an der Oberfläche sichtbar sind (vgl. Abschnitt 4.1.3).

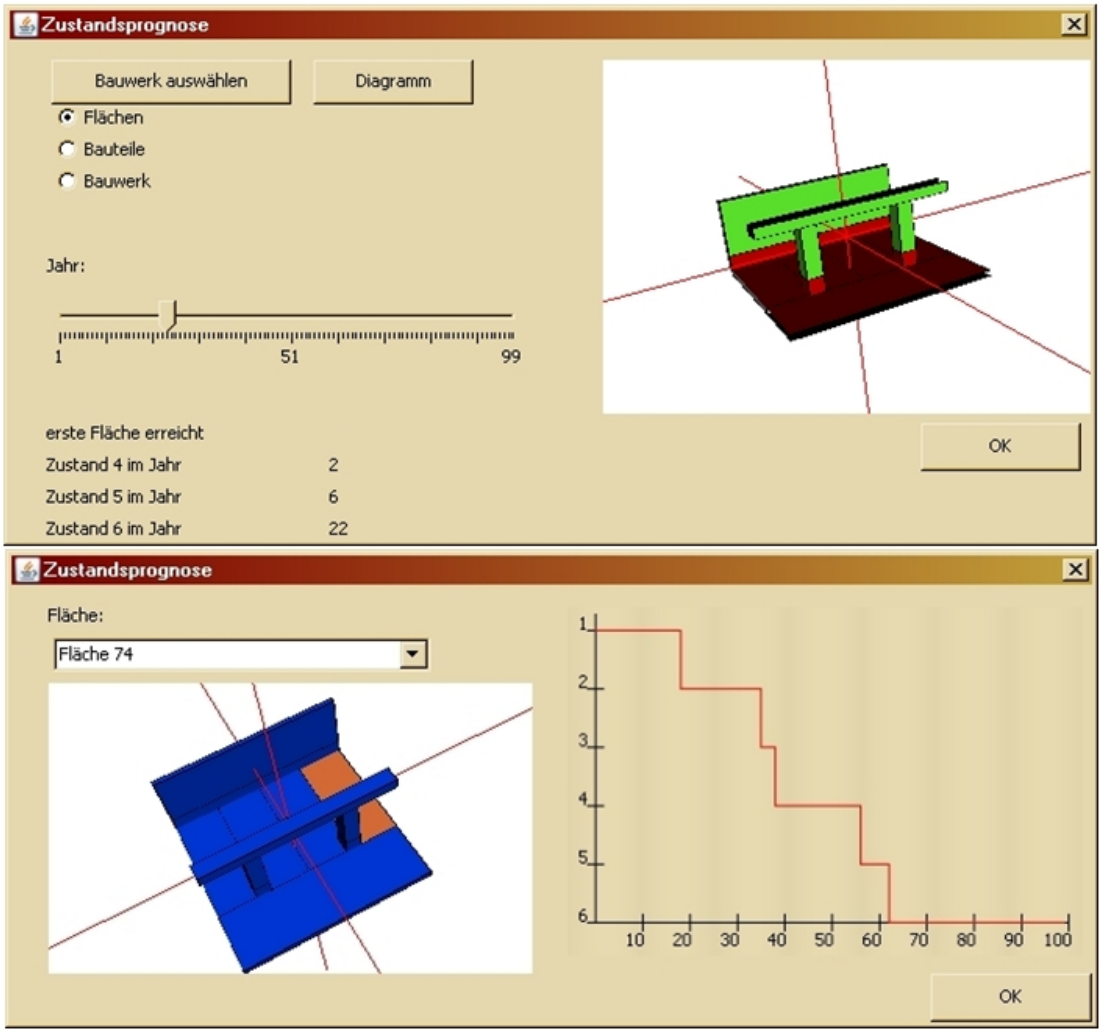

Abbildung 4.5.: Screenshot des PLMS-Prototyps: Die Zustandsnoten der einzelnen Oberflächen und Bauteile können farblich codiert für alle Zeitpunkte der Lebensdauer direkt im Bauwerksmodell (hier ein Ausschnitt aus einem Parkdeck) dargestellt werden. Eine intuitive Erfassung der Schwachstellen ist so möglich. Zusätzlich kann der Verlauf der Zustandsnote über die Lebensdauer auch detailliert in einem Diagramm dargestellt werden.

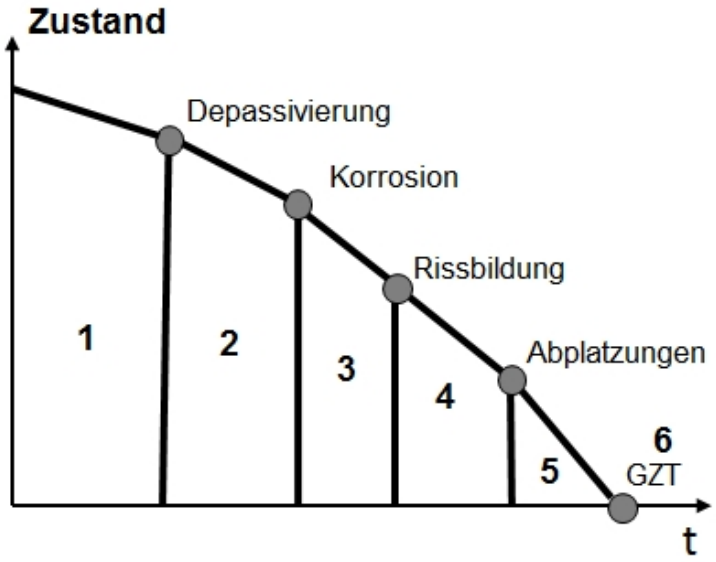

Abbildung 4.6.: Zustandsnoten nach Schießl & Mayer (2007) hier dargestellt für chlorid- und carbonatisierungsinduzierte Korrosion: Solange bis der Beton auf Höhe der Bewehrung depassiviert ist, hat das Bauteil die Note 1, bis zum Einsetzen der Korrosion danach die Note 2. Nach Beginnen der Korossion bis sich erste Risse an der Oberfläche bilden befindet sich das Bauteil in Zustand 3. Erste Abplatzungen markieren den Wechsel von Zustand 4 auf 5. Die Zustandsnote 6 ist mit Erreichen des Grenzzustands der Tragfähigkeit gegeben.

Note	*Bezeichnung*	*Beschreibung*
1	exponiert	Das Bauteil ist nicht geschützt, eine Schädigung hat jedoch noch nicht stattgefunden. Durch eine Oberflächenschutzmaßnahme kann eine Schädigung des Bauteils ausgeschlossen werden.
2	gefährdet	Das Bauteil ist nicht geschützt, eine Schädigung hat jedoch noch nicht stattgefunden. Eine Oberflächenschutzmaßnahme ist nicht mehr ausreichend, um eine Schädigung des Bauteils auszuschließen.
3	angegriffen	Das Bauteil ist einem Schädigungsmechanismus unterworfen, der jedoch noch nicht zu einer messbaren Schädigung geführt hat.
4	geschädigt	Das Bauteil ist einem Schädigungsmechanismus unterworfen, der zu einer Schädigung geführt hat. Gebrauchstauglichkeit und Tragfähigkeit werden durch die vorliegende Schädigung nicht beeinträchtigt.
5	Gebrauchsversagen	Die Gebrauchsfähigkeit des Bauteils ist durch Schädigung beeinträchtigt.
6	Tragfähigkeitsverlust	Die Tragfähigkeit des Bauteils kann nicht mit der geforderten Zuverlässigkeit sichergestellt werden.

Tabelle 4.2.: Zustandsnoten nach Schießl & Mayer (2007)

Die Zustandsnoten der einzelnen Bauteile werden aus den Zustandsnoten von deren Oberflächen, gewichtet nach deren Anteil an der Bauteiloberfläche sowie zusätzlichen Wichtungsfaktoren zur Wichtigkeit der Fläche, aggregiert. Entsprechend wird die Zustandsnote des Bauwerks aus den einzelnen Bauteilnoten bestimmt. Besonders schlechte Zustände (Noten 4-6) werden direkt von einer unteren Ebene (Fläche) auf alle höheren Ebenen (Bauteil und Gesamtbauwerk) übertragen (Schießl *et al.*, 2011).

4.1.5. Zusammenfassung

Prinzipiell kann das in dieser Arbeit vorgestellte Optimierungswerkzeug zur Instandsetzungsplanung auf Netzwerkebene mit jedem beliebigen Verfahren zum Bauwerksmanagement auf Bauwerksebene kombiniert werden, sofern dieses genaue Aussagen dazu liefert, wann ein Bauwerk spätestens instand zu setzen ist. Dieser Zeitpunkt kann dem System entweder direkt durch eine Jahreszahl oder aber implizit durch eine Schädigungsfunktion sowie einen Schwellenwert, der nicht überschritten werden darf, übergeben werden.

Dabei kann das Optimierungswerkzeug prinzipiell direkt mit einem Bauwerksmanagementsystem wie zum Beispiel dem von Zilch *et al.* (2011) oder dem PLMS aus Schießl *et al.* (2011) gekoppelt werden, so dass die für den nächsten Planungszeitraum zu berücksichtigenden Bauwerke automatisch bestimmt werden (Lukas & Borrmann, 2012). Ein solches Vorgehen setzt allerdings im hohen Maße voraus, dass objektive Kriterien existieren, welche Bauwerke in dieser Auswahl zu berücksichtigen sind: Sind nur Bauwerke zu berücksichtigen, die bis zu einem bestimmten zukünftigen Jahr „kritisch“ werden? Welches Jahr soll dabei als Schwelle dienen? Oder soll die Entscheidung über die Zustandsnote fallen? Sollen prinzipiell die 50 (100, 200) schlecht bewertetsten Bauwerke berücksichtigt werden?

In der Praxis wird vermutlich, basierend auf der Ingenieurerfahrung der Planer, eine Entscheidung zur Priorisierung einzelner Bauwerke getroffen, die auf einer Kombination dieser angesprochenen Konzepte beruht. Eine automatisierte Erstellung einer Liste der zu berücksichtigenden Bauwerke, die den Anforderungen der Praxis entspricht, ist ohne weitere Forschung in diese Richtung nicht möglich. Es ist allerdings fraglich, ob ein solches System überhaupt wünschenswert ist, da das gesamte Hintergrundwissen der Planer, das in eine solche Entscheidung eingeht, darin nur schwer abzubilden wäre.

Basierend auf diesen Überlegungen wird hier vorgeschlagen, dem Optimierungswerkzeug eine von Planungsexperten manuell erstellte Bauwerksliste zu übergeben. Die auf Bauwerksebene arbeitenden Managementsysteme können dabei jedoch herangezogen werden, um den Planern Vorschläge zu machen oder eine Vorauswahl zu treffen (z.B. alle Bauwerke, die innerhalb des Planungshorizonts „kritisch“ werden).

4.2. Modellierung von Straßenverkehr

Die Modellierung und Simulation von Verkehr ist ein äußerst vielschichtiges und komplexes Thema. Im Rahmen dieser Arbeit kann es nur oberflächlich behandelt werden.

Eine Verkehrssimulation wird hier verwendet, um den Einfluss, den verschiedene Instandsetzungsszenarien auf den Straßenverkehr haben, möglich wirklichkeitsnah zu erfassen. Im Folgenden wird kurz beschrieben, wie Verkehr entsteht, wie sich Verkehrsmengen aus bekannten Daten ableiten oder für die Zukunft prognostizieren lassen, wie sich die Verteilung des Verkehrs auf ein gegebenes Netz simulieren lässt und wie sich daraus Verkehrsbehinderungen ableiten lassen.

4.2.1. Entstehung von Verkehr

Bevor hier beschrieben wird, wie Verkehr modelliert und simuliert wird, sind noch ein paar grundsätzliche Anmerkungen dazu, was Verkehr eigentlich ist und wie er entsteht, nötig. Im Allgemeinen kann man unterscheiden zwischen Personen- und Güterverkehr. Der Einfachheit halber wird in dieser Arbeit nur Personenverkehr betrachtet, Güterverkehr lässt sich aber mit ähnlichen Modellen abbilden.

Personenverkehr ist die Menge aller Ortsveränderungen von Personen. Er entsteht dann, wenn sich die verschiedenen Bedürfnisse von Personen (z.B. Wohnen, Arbeiten, Schule, Einkaufen,...) nicht am selben Ort befriedigen lassen. Aus den verschiedenen Bedürfnissen oder Aktivitäten werden häufig Wegeketten, z.B. Wohnen - Schule - Wohnen oder Wohnen - Arbeit - Einkaufen - Wohnen, gebildet (Wermuth, 2005).

Dieses Bedürfnis nach Aktivitäten bildet die Nachfrageseite des Verkehrsmodells. Inwieweit daraus tatsächliche Verkehrsströme entstehen und wie diese sich verteilen hängt von der Angebotsseite ab. Diese wird für den öffentlichen Verkehr (Zug, Bus,...) bestimmt durch das Liniennetz und den Fahrplan, für den Individualverkehr (PKW) durch das Straßennetz, dessen Auslastung und Kapazitäten.

Im Rahmen dieser Arbeit wird nur der Individualverkehr betrachtet. Störungen, die aus den Instandsetzungsarbeiten für den öffentlichen Nahbverkehr entstehen können, werden hier nicht berücksichtigt.

4.2.2. Kategorisierung von Verkehrsmodellen

Verkehrssimulatoren lassen sich prinzipiell danach unterteilen, ob das Verhalten einzelner Fahrzeuge oder der gesamte Verkehrsfluss betrachtet wird:

Wird jedes Fahrzeug einzeln modelliert, so spricht man von einem *mikroskopischen* Modell. In diesen Modellen kann das individuelle Verhalten der einzelnen Fahrzeuge abgebildet und so ihr jeweiliger Einfluss auf den Gesamtverkehr erfasst werden. Vorteilhaft ist dies vor allem bei der Analyse sehr lokaler Szenarien, beispielsweise bei der Erfassung der lokalen Auswirkungen bei der Einführung einer Busspur. Ein Beispiel für einen mikroskopischen Verkehrssimulator ist VISSIM der ptv AG (PTV AG, 2011). Seit einigen Jahren existieren auch „submikroskopische“ Simulatoren, in denen eine zusätzliche Unterteilung in Fahrzeug und Fahrer stattfindet. Damit lassen sich auch Fahrgemeinschaften oder Fahrzeugwechsel beim Park&Ride modellieren. Ein Beispiel für einen derartigen Verkehrssimulator ist der Forschungscode MatSim (Balmer *et al.*, 2008).

Die feine Auflösung der mikroskopischen Modelle führt jedoch zu einem erhöhten Rechenaufwand. Aus diesem Grund werden in der Praxis zur Simulation großer Netze *makroskopische* Simulatoren verwendet. In diesen wird der Verkehr als kontinuierlicher Fluss, ähnlich einem Wasserfluss durch ein Rohrnetz, modelliert. Individuelle, kleinteilige Auswirkungen lassen sich damit nicht erfassen, dafür kann die Verteilung des Verkehrs auf das Straßennetz recht gut abgebildet werden.

Eine Zwischenstufe bilden die *mesoskopischen* Simulatoren. Bei diesen wird das Verhalten der Fahrzeuge an den Knotenpunkten (Kreuzungen) mikroskopisch modelliert, der Verkehr auf freier Strecke dagegen makroskopisch erfasst. Dies hat den Vorteil von einem geringerem Rechenaufwand gegenüber einem rein mikroskopischen Ansatz, bei gleichzeitiger feiner Auflösung von Abbiegevorgängen. Ein Beispiel für einen derartigen Simulator bildet das in dieser Arbeit verwendete VISUM (PTV AG, 2009b).

4.2.3. Modellbildung

Der erste Schritt zur Modellierung ist die Abgrenzung eines finiten Untersuchungsgebiets aus dem gesamten Verkehrsnetz. Dieses Gebiet wird in der Regel größer gewählt als der eigentliche Planungsraum, in dem die eigentlichen Maßnahmen[5] stattfinden, um Einflüsse des von außen kommenden (Zielverkehr) und des nach außen gerichteten (Quellverkehr) Verkehrs sowie des Durchgangsverkehrs möglichst gering gegenüber dem Binnenverkehr zu halten (vgl. Abbildung 4.7, Wermuth (2005)).

Als weitere Vereinfachung wird nicht mehr jedes einzelne Gebäude als Quelle oder Ziel betrachtet. Stattdessen wird das Untersuchungsgebiet in einzelnen Verkehrszellen zerlegt. Diese sollten so klein gewählt werden, dass der Binnenverkehr in ihnen vernachlässigbar ist.

Das Straßennetz wird als gerichteter gewichteter Graph modelliert. Die Kanten stellen dabei die Streckenabschnitte dar. Die Kantengewichte beschreiben den

[5] In dem hier behandelten Fall die Instandsetzungsarbeiten

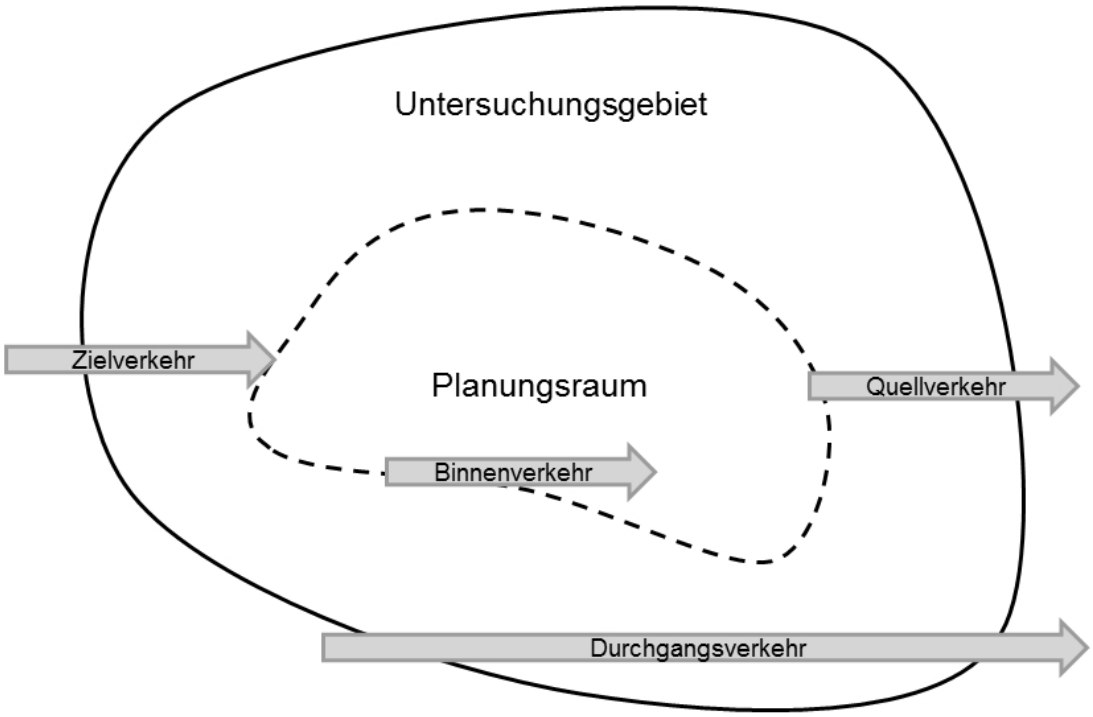

Abbildung 4.7.: Untersuchungsgebiet und Planungsraum sowie die darin auftretenden Verkehrsströme (nach Braun & Wermuth (1973))

Widerstand eines Streckenabschnitts (im Individualverkehr meistens die Fahrtdauer). Die Knoten beschreiben die Verkehrsknotenpunkte, also Kreuzungen. Zur genauen Modellierung von Abbiegebeziehungen, also auch um abzubilden, ob ein bestimmter Abbiegevorgang überhaupt erlaubt ist, müssen die Knoten in Hilfsknoten unterteilt werden (Braun & Wermuth, 1973).

4.2.4. Simulation

Für die Simulation des Verkehrs wird zunächst angenommen, dass jeder Verkehrsteilnehmer versucht, die für ihn optimale Strecke, also die mit dem geringsten Widerstand, zu wählen. Dies ist insofern eine Vereinfachung, dass dabei voraus gesetzt wird, dass jeder Verkehrsteilnehmer ein umfassendes Wissen über das Netz und seine momentane Belastung mitbringt, was in der Realität nicht gegeben ist. Außerdem wird auf diese Weise nicht berücksichtigt, dass geringfügige Unterschiede im Widerstand verschiedener Routen von den Nutzern eventuell gar nicht wahrgenommen werden. Eine Lösung, die näher an der Realität liegt ist durch stochastische Umlegungsverfahren zu erreichen (vgl. Abschnitt 4.2.5.4).

Von Bedeutung für die Verwendung des Verkehrs als Zielfunktion bei der Optimierung ist dabei, dass die Verkehrsteilnehmer jeweils nur ihr eigenes Nutzeroptimum im Blick haben, nicht aber das Systemoptimum: Würden einige Teilnehmer al-

truistisch eine für sie schlechtere Route wählen, könnte das Gesamtsystem dabei gewinnen (Heydecker, 1986; Bell & Iida, 1997). Im Extremfall kann dies sogar zu dem Paradoxon von Braess (1968) führen, dass durch die Sperrung einer Route die Fahrzeiten über das gesamte Netz geringer werden.[6] Dies zeigt, dass die Zielfunktion in dem hier behandelten Netz nicht nur nicht linear ist, sondern sich gänzlich unvorhersehbar verhalten kann. Die Optimierung wird dadurch zusätzlich erschwert.

4.2.5. Der Vier-Stufen-Algorithmus

In der Praxis wird meistens der Vier-Stufen-Algorithmus verwendet, um die Verkehrsnachfrage auf dem Netz zu verteilen. Prinzipiell ist auch möglich, die vier Stufen simultan ablaufen zu lassen, was den realen Entscheidungsvorgängen der Verkehrsteilnehmer näher kommt. Ein Beispiel für ein solches Modell findet sich in Dugge (2006). Derartige Modelle haben sich bisher aber nicht durchsetzen können.

Der Vier-Stufen-Agorithmus gliedert die Verkehrssimulation in folgende Schritte (vgl. auch Wermuth (2005)):

- Verkehrserzeugung
- Verkehrsverteilung
- Verkehrsmittelwahl / Modal-Split
- Verkehrsumlegung

In einem neueren Ansatz, dem EVA-Modell von Lohse *et al.* (1997), werden die letzten beiden Punkte, Verkehrsmittelwahl und Verkehrsumlegung, simultan ausgeführt.

Für diese Arbeit relevant ist nur der letzte Punkt, die Verkehrsumlegung. Der Vollständigkeit halber seien die anderen Punkte hier kurz beschrieben. Für eine umfassende Ausführung siehe Wermuth (2005) und Garber & Hoel (2010).

4.2.5.1. Verkehrserzeugung

In der Stufe der Verkehrserzeugung wird für jede Verkehrszelle des Untersuchungsgebietes der Quell- und Zielverkehr bestimmt. Da vereinfacht davon augegangen werden kann, dass im Laufe von 24 Stunden jeder Weg wieder nach Hause zurück führt, können Wohnungen als Quellen, die sonstigen Orte für Aktivitäten als Ziele betrachtet werden.

[6] Derartige Effekte wurden mit Aufkommen von Navigationsgeräten und TMC zusätzlich verstärkt. Rinke (2010) beschäftigt sich mit der Anwendung von Genetischen Algorithmen zur Entschärfung dieser Problematik.

Zur Ermittlung des von einer Zelle erzeugten Verkehrs, werden Daten zur soziodemografischen Struktur der Zelle (z.B. Haushaltseinkommen, Haushaltsgrößen,...) sowie zu den in ihr vorhandenen Attraktoren (Arbeitsplätze, Verkaufsfläche,...) herangezogen. Aus den soziodemografischen Daten können die in dieser Zelle durch Wohnungen erzeugten Wege ermittelt werden, indem den einzelnen Bevölkerungsgruppen typische Zahlen für pro Tag stattfindende Wege zugeordnet werden. Kalibrierte Modelle dieser Art gibt es von Kutter (1972) und Schmiedel (1983). Den Attraktoren werden auf ähnliche Weise Werte zugeordnet, die aus der Fläche (für z.B. Supermärkte, Kaufhäuser, Freizeiteinrichtungen) oder aus den vorhandenen Arbeitsplätzen (Firmen) oder Schüler- bzw. Studentenzahlen (Ausbildungseinrichtungen) ermittelt werden.

Modernere Ansätze modellieren die Wege ausgehend von Aktivitätenketten, die wiederum den verschiedenen Bevölkerungsgruppen aus soziodemografischen Erhebungen zugeordnet werden. Ein Beispiel für ein solches Modell findet sich in Hertkorn (2004). Auch für den hier verwendeteten Verkehrssimulator VISUM existiert ein Modul, VISEM, das die Verkehrserzeugung auf diese Weise modellieren kann (PTV AG, 2009a).

4.2.5.2. Verkehrsverteilung

Bei der Verkehrsverteilung werden Quell-Ziel-Beziehungen zwischen den einzelnen Verkehrszellen erstellt. Grundlage bilden dabei die in der Stufe der Verkehrserzeugung ermittelten Werte für die von den Zellen erzeugten Verkehrsmengen. Ziel hierbei ist, dem Quellverkehr jeder Zelle jeweils Ziele zuzuordnen.

Einer der meistverwendeten Ansätze hierfür ist das Gravitationsmodell (vgl. Wermuth (2005)). In Anlehnung an das Gravitationsmodell der Mechanik nimmt die Attraktivität einer Zielzelle ab, je größer der Widerstand, also je schlechter die Erreichbarkeit, zwischen den beiden Zellen ist. Der Verkehr der Quellzelle verteilt sich dann proportional zu dieser Attraktivität auf die verschiedenen Zielzellen. Je nachdem, wie sicher die Mengenangaben sind, kann das Gravitationsmodell so angewandt werden, dass nur die Quellwerte den in Stufe 1 bestimmten Werten entsprechen (quellgekoppeltes Gravitationsmodell) oder aber Quell- und Zielverkehr festgelegt sind[7] (quell- und zielgekoppeltes Gravitationsmodell).

Eine andere Möglichkeiten zur Modellierung der Verkehrsverteilung ist zum Beispiel das Entropiemodell (Wilson, 1967), bei dem die Verkehrsströme so auf das Netz verteilt werden, dass die Entropie im Gesamtsystem maximiert wird. Die Entropie steht hier in Anlehnung an den thermodynamischen Entropiebegriff für die „Unordnung“ im System. Auf den Verkehr bezogen bedeutet eine maximale

[7] Da die demografischen Daten, die die Grundlage zur Quellverkehrermittlung bilden, als zuverlässiger angesehen werden als die Strukturdaten für den Zielverkehr, wird letzteres Verfahren nur dann angewandt, wenn sehr genaue Daten z.B. zu den vorhandenen Arbeitsplätzen der Zielzellen vorhanden sind.

Entropie die gleichmäßigste Verteilung der Fahrzeuge auf das Netz, die möglich ist.

4.2.5.3. Modal-Split

In der Stufe des Modal-Split werden die in der Verkehrsverteilung erstellten Quell-Ziel-Beziehungen auf Verkehrsmittel, also Fußweg, öffentlicher Verkehr (Bus, Straßenbahn, S- und U-Bahn) und Individualverkehr (PKW und auch Fahrrad), aufgeteilt, wobei Fußweg und Fahrrad bei weiten Strecken vernachlässigbar sind. Im Rahmen dieser Arbeit wird vereinfacht davon ausgegangen, dass keine einzelnen Verkehrsteilnehmer durch eine Baustelle dazu gebracht werden, für einen Weg, den sie sonst mit dem Auto zurücklegen, auf öffentliche Verkehrsmittel umzusteigen. Damit ist auch diese Stufe für das Optimierungsproblem nicht relevant.

In die Wahl des Verkehrsmittels gehen verschiedene Faktoren ein. Einerseits gibt es Personengruppen, die an ein bestimmtes Verkehrsmittel gebunden sind, z.B. Personen, die kein Auto oder keinen Führerschein besitzen, an den öffentlichen Verkehr oder Personen, die einen schweren oder sperrigen Gegenstand transportieren müssen, an das Auto.

Für die Verkehrsteilnehmer, die an kein spezifisches Verkehrsmittel gebunden sind, gehen in die Wahl des Verkehrsmittels verschiedene objektive (z.B. Kosten, Erreichbarkeit, Fahrtzeit, Parkplatzverfügbarkeit am Zielort,...) und subjektive (z.B. Abneigung gegen ein bestimmtes Verkehrsmittel aus Gründen der Bequemlichkeit oder aus Umweltbewußtsein) Gründe ein. Zur Modellierung dieser Wahl können, wenn vorhanden, Daten aus Haushaltsbefragungen herangezogen werden (Wermuth, 2005). Sind keine solchen Daten bekannt, kann das Verhalten der Bewohner eines Gebiets, wie schon bei der Verkehrserzeugung (Absatz 4.2.5.1), aus soziodemografischen Daten abgeleitet werden. Die letztendliche Aufteilung erfolgt dann über ein probabilistisches Modell basierend auf dem relativen Nutzen einer Wahl (Probit-Modell und Logit-Modell).

4.2.5.4. Verkehrsumlegung

Bei der Verkehrsumlegung werden die zuvor ermittelten Verkehrsströme zwischen den einzelnen Verkehrszellen auf die verschiedenen möglichen Routen im Netz verteilt, um die tatsächliche Belastung der einzelnen Netzelemente zu ermitteln. Dies ist der Teil der Verkehrssimulation, der für die Optimierung der Instandsetzungszeitpläne von Bedeutung ist (wenn man davon ausgeht, dass die Verkehrsbehinderungen durch die Maßnahmen keinen Einfluss auf die Verkehrsmittelwahl oder gar die Zielwahl haben).

Für die Umlegung gibt es verschiedene mögliche Verfahren, die auch alle in der Praxis Verwendung finden. In der Version 11 des kommerziellen Verkehrssimu-

lators VISUM sind neun verschiedene Umlegungsmethoden implementiert (PTV AG, 2009b). Die Umlegungsmethoden lassen sich auf verschiedene Weise kategorisieren:

Die erste Unterscheidungsmöglichkeit ist die Unterteilung in statische und dynamische Methoden: Während bei den statischen Methoden eine konstante Verkehrsnachfrage innerhalb des Betrachtungszeitraumes (z.B. der Spitzenstunde) angenommen wird, kann in dynamischen Umlegeverfahren die veränderliche Nachfrage über den Tag (Tagesganglinie) berücksichtigt werden und in die Berechnung eingehen (Wermuth, 2005).

Die zweite Unterteilung erfolgt in deterministische und stochastische Verfahren. Bei einer deterministischen Umlegung wird für jeden Nutzer angenommen, dass er seine persönliche Route optimiert. Dieses Nutzeroptimum ist dann gegeben, wenn der Widerstand auf allen alternativen Routen gleich ist[8] (1. Wardrop'sches Prinzip, Wardrop (1952)). Wie bereits unter 4.2.4 erwähnt, ist dies keine realitätsnahe Annahme, da sie ein vollständiges Wissen jedes Beteiligten über das gesamte Netz und seine Widerstände voraussetzt. Um das Unwissen und die unterschiedliche subjektive Einschätzung der einzelnen Nutzer abzubilden, werden in stochastischen Umlegungsverfahren die einzelnen Routen für jeden Nutzer mit einem eigenen zufälligen Schätzfehler versehen.

Andere Umlegungsverfahren berücksichtigen zusätzlich den Rückstau, der an Lichtsignalanlagen entsteht (WDDTA, Bellei *et al.* (2005)), oder die Wirkung von Mautsystemen auf die Routenwahl (TRIBUT, PTV AG (2009a)).

Am besten spiegeln dynamische, stochastische Verfahren die Realität wieder. Leider können diese aus verschiedenen Gründen nicht für die hier vorgestellte Optimierung verwendet werden. Das Problem bei stochastischen Verfahren ist, dass sie für dasselbe Netz in verschiedenen Läufen verschiedene Ergebnisse produzieren können. Diese Abweichungen sind so klein, dass sie normalerweise vernachlässigbar sind - bei den hier zur Optimierung verwendeten Metaheuristiken ist es jedoch essentiell, dass die gleiche Lösung auch immer genau den gleichen Zielfunktionswert zugewiesen bekommt. Daher scheiden stochastische Verfahren hier aus.

Ob statische oder dynamische Verfahren verwendet werden, macht im Hinblick auf die Optimierung eigentlich keinen Unterschied. Dynamische Verfahren bedeuten aber einen erhöhten Rechenaufwand und damit eine längere Rechenzeit. Bei den mehreren Tausend Aufrufen der Simulation, die ein Optimierungsdurchlauf benötigt, bedeutet die Verwendung von statischen Verfahren eine hohe Zeitersparnis.

Zur Verwendung in dieser Arbeit wurde die *Gleichgewichtsumlegung* (Beckmann *et al.*, 1956) gewählt. Diese hat die Vorteile, dass für sie ein eindeutiges Ergebnis

[8] Zur Definition des Widerstandes einer Route wird meistens ihre Fahrtzeit herangezogen. Diese wird beschrieben durch eine mit zunehmender Belastung der Strecke monoton steigende Funktion, definiert durch Streckenlänge, Kapazität und zulässige Höchstgeschwindigkeit.

existiert und dieses für ein statisches Verfahren relativ wirklichkeitsnah ist (PTV AG, 2009a). Prinzipiell ließe sich das beschriebene Optimierungsverfahren jedoch mit jedem beliebigen deterministischen Umlegungsverfahren koppeln.

Basierend auf dem oben erwähnten 1. Wardrop'schen Prinzip wird bei der Gleichgewichtsumlegung das Problem der Umlegung als Optimierungsproblem umformuliert: Es wird nach der Verteilung gesucht, bei der das Integral über die Widerstände aller Kanten im Netz minimal ist. Als Randbedingungen gehen dabei die in den Stufen Verkehrserzeugung und Verkehrsverteilung ermittelten Verkehrsströme ein, das heißt, alle Fahrzeuge, die von Zelle i zu Zelle j fahren, müssen für alle Zellen i und j auch wirklich so im Ergebnis auftauchen. Außerdem gilt entsprechend der Kichhoff'schen Regel, dass in jedem Knoten die Summe der einfahrenden Ströme gleich der der ausfahrenden Ströme sein muss. Zudem müssen die Belastungen natürlich positiv sein, wobei sich die Belastung einer Kante als die Summe der Belastungen aller Wege ergibt, die über sie führen.

Gelöst wird dieses Problem iterativ: Als Startlösung wird das Ergebnis einer anderen Umlegung verwendet, beispielsweise einer Sukzessivumlegung, bei der der Verkehr stufenweise auf ein zuvor leeres Netz verteilt wird. Ausgehend von dieser Startlösung läuft die Optimierung nun in zwei Schleifen ab.

In der inneren Schleife wird die Belastung über das Netz durch paarweisen Vergleich jeweils zweier Routen so ausgeglichen, dass der Widerstand über alle Routen gleich ist (Netzausgleich). Die äußere Schleife überprüft, ob durch den Netzausgleich neue Routen mit geringeren Widerständen entstanden sind. Ist dies der Fall, so wird der Netzausgleich wiederholt, sonst ist der Gleichgewichtszustand und damit die Lösung erreicht.

4.2.6. Zusammenfassung

In diesem Abschnitt wurde beschrieben, wie Straßenverkehr modelliert und simuliert werden kann. Als typisches Vorgehen wurde der Vier-Stufen-Algorithmus beschrieben. Für diese Arbeit als Eingangsgröße für die Optimierung von Bedeutung ist vor allem dessen vierte Stufe, die Umlegung. Eventuell existierende Einflüsse der Instandsetzungsarbeiten auf die anderen Stufen werden hier vernachlässigt.

Für die Verkehrsumlegung, also die Verteilung des Verkehrs auf das Straßennetz, kommen verschiedene Ansätze in Frage. Zur Kopplung mit einem Optimierungslauf muss dabei gewährleistet sein, dass bei gleichen Randbedingungen (gleiche Verkehrsnachfrage und Störungen an denselben Stellen) die Verkehrsumlegung dasselbe Ergebnis liefert. Zudem sollte die Berechnung möglichst schnell gehen, da die Simulation während eines Optimierungslaufes mehrere Tausend Mal aufgerufen wird.

Diese Voraussetzungen sind mit allen deterministischen, statischen Umlegungsverfahren gegeben. Im Rahmen dieser Arbeit fiel die Wahl auf die Verwendung der Gleichgewichtsumlegung, eine Kopplung mit anderen deterministischen Verfahren ist aber ebenso denkbar.

5. Genetische Algorithmen

5.1. Grundlagen

Genetische Algorithmen (GA) bilden einen Teilbereich der Evolutionären Algorithmen. Deren Idee basiert auf der Evolutionstheorie von Charles Darwin (Darwin, 1859), wonach sich Arten durch Kreuzung, zufällige Mutation und natürliche Auslese entsprechend der Anpassung an den jeweiligen Lebensraum bilden. Entwickelt wurden Genetische Algorithmen in den 1970ern durch Holland (1975). Dabei ging es ursprünglich gar nicht um die Entwicklung einer Optimierungsmethode sondern um die Untersuchung der Funktionsweise der natürlichen Anpassungsmechanismen(vgl. De Jong (1993)).

Dessen ungeachtet zeigen sie sich als gut geeignet zur Lösung von Optimierungsproblemen. Die Idee dahinter ist, das Optimierungsproblem als die Umgebung zu betrachten, an die sich eine Population möglicher Lösungen, *Individuen* genannt, anpassen muss. Der Wert der Zielfunktion liefert dann ein Maß für die Güte dieser Anpassung. Man spricht in diesem Zusammenhang auch von der *Fitness* einer Lösung bzw. eines Individuums. Für die Zielfunktion wird häufig auch der Begriff *Fitnessfunktion* gebraucht. Bei einem Maximierungsproblem haben die Individuen mit höheren Werten für die Fitnessfunktion die höhere Fitness (also die bessere Anpassung an die „Umwelt“), bei Minimierungsproblemen, diejenigen mit niedrigeren Werten.

5.2. Algorithmus

Genetische Algorithmen sind im Grunde genommen keine starren Algorithmen als vielmehr eine Art Bausatz aus kleineren Algorithmusstücken (vgl. Eiben & Smith (2003)). Zur Lösung eines Problems mit Genetischen Algorithmen müssen diese Bausteine zunächst auf das Problem abgestimmt zusammen gesetzt werden.

Beispiel: *Zur Veranschaulichung der einzelnen Schritte des Algorithmus wird im Folgenden als Beispiel ein einfaches Rucksackproblem (siehe Abschnitt 2.2) verwendet. Gewicht und Nutzwert der fünf Gegenstände sind in Tabelle 5.1 gegeben. Der zugehörige Rucksack kann ein Gewicht von 15 Einheiten fassen.*

Nummer	Gewicht	Nutzen
1	6	74
2	6	10
3	9	12
4	1	1
5	3	51

Tabelle 5.1.: Eingangsdaten des beispielhaften Rucksackproblems

5.2.1. Lösungsrepräsentation

Zur Anwendung von Genetischen Algorithmen auf ein Optimierungsproblem werden die möglichen Lösungen des Problems zunächst codiert. Dies erfolgt in Anlehnung an das natürliche Vorbild: Die Erbinformation eines Lebewesens ist in seinen Genen als Abfolge von vier verschiedenen Aminosäuren codiert. Mehrere Gene zusammen bilden ein *Chromosom*. Auch bei den künstlichen Individuen des Algorithmus spricht man daher bei ihrer codierten Version von ihrem *Genotyp* bzw. *Chromosom*. Diese Chromosomen sind dann wiederum aus einzelnen Genen zusammengesetzt, die jeweils die einzelnen Eigenschaften des Individuums codieren. Die Lösung selbst in ihrer uncodierten Form ist dann der *Phenotyp*.

Bei der Codierung ist darauf zu achten, dass jeder Genotyp genau einem Phenotyp entspricht. Ein und der selbe Phenotyp kann aber durchaus durch mehrere verschiedene Genotypen beschrieben werden. Die Art der Codierung ist aus Sicht des Algorithmus beliebig, allerdings hat sie großen Einfluss auf die Leistungsfähigkeit des Algorithmus und sollte in Abstimmung mit dem zu lösenden Problem gewählt werden (Eiben & Smith, 2003).

Eine der ältesten Möglichkeiten, Lösungen zu codieren ist die Binäre Repräsentation (Holland, 1975). Dabei werden die Lösungen als einfache Bitstrings, also Strings von Nullen und Einsen beschrieben. Andere klassische Repräsentationen sind die Repräsentation als String von Ganzzahl- oder auch Gleitkommavariablen, sowie die als Permutation der Ganzzahlen von 0 bis $n-1$[1].Auch komplexere Repräsentationen sind denkbar: So beschreiben Michalewicz & Fogel (2004) zum Beispiel auch die Möglichkeit einer Zustandsmaschine oder eines Baums von Symbolausdrücken.

Beispiel fortgeführt: *Eine mögliche (nicht die ideale) Repräsentation des zuvor beschriebenen Rucksackproblems beschreibt die Lösungen als Binärstring der Länge fünf. Eine 1 an i-ter Stelle des Strings, bedeutet, dass der Gegenstand mit der Nummer i im Rucksack ist. Der String* 01100 *steht zum Beispiel für die Lösung, bei der nur die Gegenstände mit den Nummern 2 und 3 im Rucksack sind.*

[1]Eine Permutation unterscheidet sich von einem Ganzzahlstring dadurch, das jedes Element in ihr genau einmal vorkommt

5.2.2. Ablauf der Optimierung

Ein Genetischer Algorithmus folgt im wesentlichen den folgenden Schritten:

1. **Selektion der Eltern**: Aus der gesamten Lösungspopulation der Größe ψ wird eine Untermenge der Größe μ, die Elternmenge, ausgewählt. Die in ihr befindlichen Individuen sind zur Fortpflanzung bestimmt.
2. **Kreuzung**: Jeweils n (normalerweise 2) Individuen aus der Elternmenge werden mit der Wahrscheinlichkeit $p_{\text{crossover}}$ miteinander gekreuzt. Dabei entstehen 1 bis n neue Individuen, die Kinder.
3. **Mutation**: Mit einer Wahrscheinlichkeit p_{mutation} erfolgt eine zufällige Änderung an den neuen Individuen.
4. **Selektion der Überlebenden**: Aus der Gesamtmenge der alten Generation und der neuen Individuen werden nur so viele ausgewählt, dass die Populationsgröße wieder ψ entspricht[2].

Die einzelnen Schritte werden im Folgenden näher behandelt.

5.2.2.1. Selektion der Eltern

Die Auswahl der möglichen Eltern erfolgt probabilistisch nach deren Fitnesswerten, also dem Wert, den eine Lösung beim Auswerten der Fitnessfunktion erhält. Die Elternmenge kann dabei mehrere Kopien desselben Individuums aus der Gesamtpopulation enthalten. Im Wesentlichen gibt es dabei drei Möglichkeiten der Selektion:

Bei der *Fitness-proportionalen Selektion* (Holland, 1975) wird jedem Individuum x der Population eine Wahrscheinlichkeit zugeordnet, die proportional zu dessen Fitnesswert f_x ist. Am einfachsten geschieht dies über

$$p_{\text{select}}(x) = f_x / \sum_{j=1}^{\psi} f_j \tag{5.1}$$

Dies hat allerdings den Nachteil, dass herausragend gute Individuen eine übermäßig hohe Wahrscheinlichkeit haben, gewählt zu werden, was zu einer Übernahme des gesamten Genpools und damit zu vorzeitiger Konvergenz[3] führen kann. Andererseits findet bei nahe beieinander liegenden Fitnesswerten kein ausreichender Selektionsdruck mehr statt. Ein weiterer negativer Effekt ist, dass es sich auf

[2] Es gibt auch Strategien, bei denen nur die jeweiligen Eltern eines neu erzeugten Individuums mit diesem in Konkurrenz treten. Eine solche beschreibt Bartlett (1995) unter dem Namen *Weak Parent*. Dabei wird das schwächere Elternteil durch das fittere Kind ersetzt. Das schwächere Kind wird verworfen.

[3] Der Algorithmus konvergiert dabei nicht auf das globale sondern auf ein zufälliges lokales Optimum.

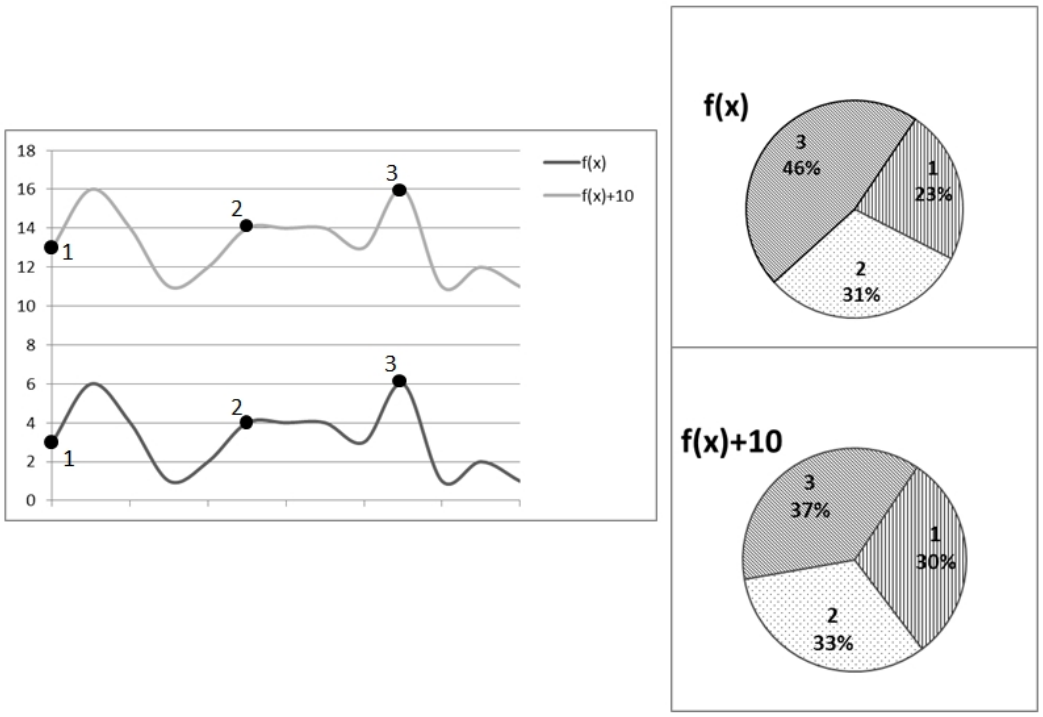

Abbildung 5.1.: Auswirkung der linearen Transposition auf die Auswahlwahrscheinlichkeiten bei der Fitness-proportionalen Selektion: Links dargestellt sind die Fitnessfunktionen $f(x)$ und $f'(x) = f(x) + 10$. Die Tortendiagramme rechts zeigen die Auswahlwahrscheinlichkeiten in einer Population aus den drei markierten Individuen 1, 2 und 3, oben für die Fitnessfunktion $f(x)$ unten für $f'(x)$. Während sich diese Auswahlwahrscheinlichkeiten für $f(x)$ stark unterscheiden, sind sie für $f'(x)$ fast gleich. (Abbildung modifiziert nach Eiben & Smith (2003))

die Optimierung auswirkt, wenn die Fitnessfunktion linear transponiert wird, das bedeutet, dass die Auswahlwahrscheinlichkeit der einzelnen Individuen von ihren absoluten Fitnesswerten abhängt (vergleiche Abbildung 5.1).

Beispiel fortgeführt: *Für das zuvor beschriebene Rucksackproblem wird mit der gewählten Repräsentation eine Population der Größe $\psi = 5$ zufällig erstellt. Die Individuen sind in Tabelle 5.2 dargestellt. Zusätzlich ist dort die jeweilige Auswahlwahrscheinlichkeit nach Gleichung 5.1 angegeben.*

Nun wird eine Elternmenge der Größe $\mu = 4$ ausgewählt. Die letzte Spalte von Tabelle 5.2 gibt an, wie viele Exemplare des jeweiligen Individuums erwartungsgemäß in der Elternmenge enthalten sind. Tatsächlich wäre es möglich, dass beispielsweise zwei Mal das Individuum 1 und zwei Mal das Individuum 2 in die Elternmenge kopiert wird.

Eine Möglichkeit, die oben beschriebenen Probleme zu überwinden, ist die *Selektion durch Ranking* (Baker, 1987). Dabei ist die Auswahlwahrscheinlichkeit der

	Gewicht	**Nutzen**	**fitnessproportionale Auswahlwahrscheinlichkeit [%]**	**erwartete Anzahl**
00101	12	63	26.1	1.04
11000	12	84	34.9	1.39
01001	9	61	25.3	1.01
01010	7	11	4.6	0.18
01100	15	22	9.1	0.37

Tabelle 5.2.: Individuen der Startgeneration des Beispielproblems mit Auswahlwahrscheinlichkeit nach Fitnesspropoprtionaler Selektion.

einzelnen Lösungen nicht mehr abhängig vom absoluten Wert der Fitnessfunktion, sondern vom Rang, den die Lösung innerhalb der Population besitzt. Meistens geschieht dies linear: Setzt man für das beste Individuum der Population den Rang gleich $\psi - 1$ und für das schlechteste Individuum den Rang 0, dann gilt für die Auswahlwahrscheinlichkeit des Individuums mit Rang i

$$p_{\text{select}}(i) = \frac{(2-s)}{\psi} + \frac{2i(s-1)}{\psi(\psi-1)} \tag{5.2}$$

Dabei ist s ein Parameter $1.0 < s \leq 2.0$, der festlegt, wie hoch der Selektionsdruck ist (vergleiche Abbildung 5.2). Wird ein noch höherer Selektionsdruck gewünscht, kann die Zuweisung der Auswahlwahrscheinlichkeit auch nach anderen Funktionen, z.B. exponentiell, erfolgen.

Beispiel fortgeführt: *Mit der Selektion durch Ranking ergeben sich andere Auswahlwahrscheinlichkeiten für die Individuen der Startgeneration des Beispiels. Die Wahrscheinlichkeiten der Wahl sowie die Anzahl der erwarteten Kopien der einzelnen Individuen für $s = 2$ sind in Tabelle 5.3 aufgeführt. Auch hier kann die tatsächliche Zusammensetzung der Elternmenge von den erwarteten Werten abweichen. Zum Beispiel kann die Elternmenge je eine Kopie von Individuum 1 und 2 und zwei von Individuum 3 enthalten.*

Tate & Smith (1995) und Deeter & Smith (1997) beschreiben eine weitere Selektionsmethode, die das Problem der hervorstechenden Individuen umgehen soll. Auch hierbei werden die Individuen nach ihrem Rang geordnet, wobei das beste Individuum den Rang 1 erhält. Daraufhin wird eine Zufallszahl z zwischen 0 und $\sqrt{\psi}$ erzeugt. z wird daraufhin quadriert und das Ergebnis auf eine ganze Zahl abgerundet ($\lfloor z \rfloor$). Das gewählte Individuum ist dann dasjenige mit dem Rang $\lfloor z \rfloor + 1$. Durch die Parabelform werden damit Individuen mit einem niedrigen Rang bevorzugt gewählt. Über eine Änderung des Wurzelgrades lässt sich der Selektionsdruck steuern.

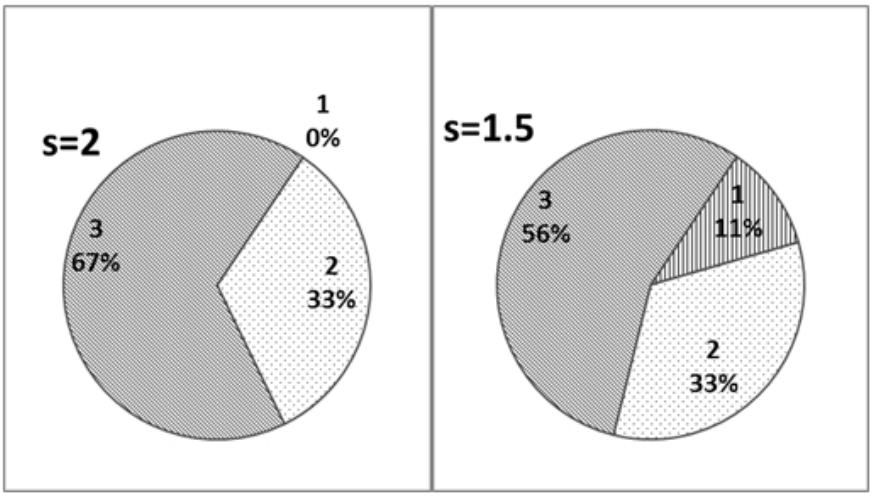

Abbildung 5.2.: Die Auswahlwahrscheinlichkeiten für die Individuen aus Abbildung 5.1 mit einer Selektion durch Ranking, links mit $s = 2$, rechts mit $s = 1.5$. Wie man sieht, hat der Parameter s einen großen Einfluss auf den Selektionsdruck. Der absolute Wert der Fitnessfunktion hat bei der Selektion durch Ranking keinen Einfluss auf die Auswahlwahrscheinlichkeit.

Der am häufigsten verwendete Selektionsmechanismus ist jedoch die *Turnierselektion*. Dabei werden jeweils zwei (oder auch q) Individuen miteinander verglichen, das bessere (bzw. das beste) dieser Individuen wird in die Elternmenge aufgenommen. Gegenüber der Selektion durch Ranking hat dies den Vorteil, dass nicht die gesamte Population sortiert werden muss. Ein weiterer Vorteil liegt darin, dass sich der Selektionsdruck leicht dadurch steuern lässt, wie viele Individuen an einem Turnier beteiligt werden, also wie groß q gewählt wird (Bäck, 1995). Bei dem üblichen binären Turnier (also $q = 2$) verhält sich der Algorithmus ähnlich wie bei der Selektion durch lineares Ranking mit $s = 2$: Eine Lösung hoher Fitness benötigt bei beiden die gleiche Zeit um die gesamte Population zu übernehmen,

	Auswahlwahrscheinlichkeit nach Ranking [%]	**erwartete Anzahl**
00101	30	1.2
11000	40	1.6
01001	20	0.8
01010	0	0
01100	10	0.4

Tabelle 5.3.: Auswahlwahrscheinlichkeit mit Selektion durch Ranking der Individuen der Startgeneration des Beispielproblems mit $s = 2$.

wie in Goldberg & Deb (1991) gezeigt. Mit zunehmendem q erhöht sich der Selektionsdruck. Bei $q = 5$ geht die Hälfte der Population verloren (Blickle & Thiele, 1996). Zusätzlich lässt sich bei der Turnierselektion auch leicht festlegen, ob das schlechteste Individuum (bzw. die $q - 1$ schlechtesten Individuen) von vornherein von der Wahl ausgeschlossen sein sollen oder nicht, indem man bei der Wahl der Turnierteilnehmer eine Mehrfachauswahl zulässt oder nicht (Eiben & Smith, 2003).

Beispiel fortgeführt: *Für die Anwendung der Turnierselektion auf das Beispielproblem müssen keine Auswahlwahrscheinlichkeiten für die einzelnen Individuen berechnet werden. $\mu = 4$ mal werden zwei Individuen der Population zufällig ausgewählt und miteinander verglichen. Im ersten Schritt werden zum Beispiel die Individuen 1 und 4 gezogen. Individuum 1 hat die höhere Fitness und wird in die Elternmenge kopiert. Im zweiten Schritt Individuen 2 und 3, Individuum 2 wird in die Elternmenge kopiert. Dritter Schritt Individuen 1 und 3, gewählt wird Individuum 1. Vierter Schritt 3 und 4, Auswahl 3. Die Elternmenge enthält nun also zwei Kopien von Individuum 1 und je eine Kopie von Individuum 2 und 3.*

5.2.2.2. Kreuzung

Im Schritt Kreuzung werden jeweils n Individuen miteinander neu kombiniert um ein bis n neue Individuen zu erzeugen. In Anlehnung an das natürliche Vorbild wird meistens $n = 2$ gewählt. Mathematisch betrachtet gibt es dafür keinen tieferen Grund, allerdings hat die Verwendung von mehr als zwei Eltern bei den meisten Problemtypen keinen Vorteil (Eiben & Smith, 2003).

Louis & Rawlins (1993) vergleichen die Kreuzung mit den Sprüngen im Simulierten Abkühlen: Auch hier nimmt die Größe der Sprünge mit Fortschritt des Optimierungsprozesses ab, da sich die Individuen der Population mit fortschreitender Konvergenz immer mehr angleichen.

Die Operatoren zur Kreuzung müssen in Abstimmung sowohl auf die gewählte Repräsentation als auch auf das Problem gewählt werden. Es gibt keinen idealen Kreuzungsoperator, der auf allen Problemen die beste Wahl ist (Fogel & Ghozeil, 1997). Radcliffe (1991) formuliert zwei Anforderungen an die Kreuzungsoperatoren: Die erste ist der sogenannte *Respekt*, was bedeutet, dass Eigenschaften[4], die beiden Eltern gemeinsam sind, auch in den Nachkommen vorkommen müssen (Ein Kind von zwei blauäugigen Eltern hat ebenfalls blaue Augen). Die zweite Anforderung an einen Kreuzungsoperator bezeichnet Radcliffe als *Proper Assortment*[5]. Dabei soll es möglich sein, dass ein Kind zweier Eltern mit kompatiblen Eigenschaften (zum Beispiel hat ein Elternteil blaue Augen, der andere braune Haare) beide diese Eigenschaften (also blaue Augen und braune Haare) aufweist.

[4] *formae*
[5] in etwa: „geeignete Sortierung“

Allerdings ist es nicht immer möglich, Operatoren zu entwerfen, die beiden Anforderungen gerecht werden.

Die Entscheidung, was genau die Eigenschaften eines Individuums ausmacht, ist stark problemabhängig. Wird beispielsweise ein Problem im Gen als Permutation codiert, so kann entweder nur die relative Ordnung der Elemente für die Lösungsgüte wichtig sein, also, dass z. B. Element x vor Element y steht, egal, wie viele Elemente dazwischen stehen. Ein Beispiel für ein solches Problem ist die Erstellung eines Bauzeitenplanes. Bei einer anderen Problemstellung dagegen, zum Beispiel beim Problem des Handlungsreisenden, ist die absolute Ordnung der Elemente wichtig, also z.B., dass x direkt vor y steht. Dies ist bei der Wahl der Operatoren zu berücksichtigen.

Beispiel fortgeführt: *Die Individuen 1 und 2 aus Tabelle 5.2 sollen miteinander gekreuzt werden. Als einfacher Kreuzungsoperator wird die Ein-Punkt-Kreuzung gewählt: Ein zufälliger Trennungspunkt wird bestimmt, z.B. nach Eintrag 3. Die Einträge vor diesem Punkt werden aus dem ersten Elternteil, Individuum 1,* 00101*, in das Kind übernommen, die Einträge danach aus dem zweiten Elternteil, Individuum 2,* 11000*. Das erzeugte Kind ist dann der String* 00100*, mit Gewicht 9 und Nutzen 12.*

5.2.2.3. Mutation

Mutation sorgt dafür, dass der gesamte Lösungsraum mit einer gewissen Wahrscheinlichkeit erreicht werden kann (Radcliffe, 1991). Man kann sie auch als asexuelle Fortpflanzung mit nur einem Elternteil betrachten.

Sie kann statt des Kreuzungsmechanismus angewendet werden. Dann werden zwei Individuen der Elternmenge nur mit einer Wahrscheinlichkeit $p_{\text{crossover}}$ miteinander gekreuzt, ansonsten werden zwei neue Individuen durch Mutation der Elterninidividuen erzeugt[6]. Eine andere Möglichkeit ist, die Muation der Kreuzung nachzuschalten: Die bei der Kreuzung entstandenen Individuen werden noch zusätzlich einer Mutation unterworfen.

Wie auch die Kreuzungsoperatoren müssen die Mutationsoperatoren auf das Problem und die Repräsentation abgestimmt werden.

Bei binärer, Ganzzahl- und Gleitkommarepräsentation wird jedes Gen des Chromosoms einzeln betrachtet und mit einer Wahrscheinlichkeit p_{mutation} abgeändert: Bei der binären Repräsentation wird der Wert eines Gens von 0 auf 1 bzw. umgekehrt geändert. Bei den anderen beiden Repräsentationen ist einerseits eine Änderung auf einen zufälligen Wert möglich. Häufiger ist es aber sinnvoll, den Wert nur in einem geringen Bereich abzuwandeln, zum Beispiel indem man eine Zufallszahl aus einer Verteilung um 0 dazu addiert. Die Mutationswahrschein-

[6] Ein anderer Teilbereich der Evolutionären Algorithmen, das Evolutionary Programming (Fogel *et al.*, 1966; Fogel, 2006), verwendet Mutation als einzigen Suchoperator.

lichkeit p_{mutation} wird im allgemeinen so gewählt, dass im Schnitt ein Gen pro Chromosom abgewandelt wird.

Da bei der Permutationsrepräsentation eine Änderung eines einzelnen Gens die Lösung ungültig machen würde, muss hier das Chromosom als ganzes mutiert werden. Ob man hierbei die Werte zweier Gene vertauscht, die Werte der Gene zwischen zwei zufälligen Punkten durcheinanderwürfelt, diese invertiert oder das Chromosom auf andere Art verändert, hängt einerseits vom Problemtyp andererseits von der gewünschten Stärke der Mutation ab.

Beispiel fortgeführt: *Das zuvor durch Kreuzung erzeugte Individuum* 00100 *wird nun einer Mutation unterworfen. Jede Stelle des Chromosoms wird mit der Wahrscheinlichkeit* $p = 0.2$ *von* 0 *zu* 1 *bzw. umgekehrt umgewandelt. dabei entsteht das Individuum* 00110 *mit dem Gewicht 10 und dem Nutzen 13.*

5.2.2.4. Selektion der Überlebenden

Nach der Erzeugung der neuen Individuen durch Kreuzung und Mutation muss die Population wieder auf ihre ursprüngliche Größe ψ reduziert werden. Auch hierzu gibt es verschiedene Strategien.

Eine Möglichkeit ist, die Individuen entsprechend ihres Alters auszuwählen. Für den Fall, dass die Größe der Elternmenge μ gleich der Populationsgröße ist, bedeutet dies, dass die gesamte Elterngeneration durch ihre Nachkommen ersetzt wird ((μ,λ)-Selektion, Coello Coello *et al.* (2007)). Im anderen Extremfall, bei nur einem erzeugten Nachkommen pro Generation, entspricht die Population dann einer FIFO-Schlange[7].

Beispiel fortgeführt: *Durch Kreuzung und Mutation wurden für das Beispielproblem die vier neuen Individuen* 00110 *(Gewicht 10, Nutzen 13),* 01101 *(18,73),* 01011 *(10,62) und* 01001 *(9,61) erzeugt*[8]*. Nach altersbasierter Auswahl werden diese vier Individuen in die nächste Generation übernommen, zusätzlich überlebt ein zufällig gewähltes Individuum der Elterngeneration, da alle diese Indieviduen gleich alt sind.*

Eine andere Strategie ist, die Überlebenden für die nächste Generation aus Eltern und Nachkommen entsprechend ihrer Fitness auszuwählen ($(\mu + \lambda)$-Selektion, Coello Coello *et al.* (2007)). Prinzipiell kommen dazu die für die Selektion der Eltern in Abschnitt 5.2.2.1 beschriebenen Verfahren in Frage. Im Gegensatz zur Auswahl der Eltern wird hier allerdings im Allgemeinen die mehrfache Auswahl des selben Individuums nicht zugelassen.

[7] *fist in - first out*: Das erste erzeugte Element, ist auch das erste, das wieder aus der Population entfernt wird.

[8] Die Lösung 01101 beschreibt eine ungültige Lösung, da das Gewichtslimit 15 des Rucksacks mit ihr überschritten wird. Zur Behandlung solcher Lösungen siehe Abschnitt 5.2.3. Für dieses Beispiel wird im Folgenden angenommen, dass diese Lösung den Nutzen 0 hat

	Gewicht	**Nutzen**	**Auswahlwahrscheinlichkeit** [%]
00101	12	63	16.7
11000	12	84	22.3
01001	9	61	16.2
01010	7	11	2.9
01100	15	22	5.8
00110	10	13	3.4
01101	18	0	0
01011	10	62	16.4
01001	9	61	16.2

Tabelle 5.4.: Die Individuen der Elterngeneration sowie die neu erzeugten Individuen des Beispielproblems mit ihrer jeweiligen Auswahlwahrscheinlichkeit.

Beispiel fortgeführt: *Aus den neun Individuen, fünf aus der Elterngeneration und vier neu erzeugte, sollen die Überlebenden nach Fitness ausgewählt werden. Die Auswahlwahrscheinlichkeiten der einzelnen Individuuen sind in Tabelle 5.4 wiedergegeben. Als Überlebende für die nächste Generation werden beispielsweise die Individuen* 11000*,* 01001*,* 01100*,* 01011 *und* 01001 *ausgewählt.*

Gemeinsam haben diese beiden Strategien, dass die beste gefundene Lösung aus der Population verloren gehen kann. Die Wahrscheinlichkeit dafür ist bei der altersbasierten Auswahl besonders hoch, doch auch bei einer fitnessbasierten Auswahl ist die Möglichkeit dazu gegeben. Eine Möglichkeit diesem Problem zu begegnen ist die Strategie des *Elitismus*. Dabei wird das beste Individuum der Population auf jeden Fall in die nächste Generation kopiert. Erst wenn eine bessere Lösung gefunden wurde verliert es seinen Elitestatus an diese.

In Eiben & Smith (2003) wird auch noch die Strategie erwähnt, deterministisch die jeweils μ schlechtesten Individuen aus der Population zu entfernen. Dies kann jedoch durch die Reduzierung der Diversität in der Population zu verfrühter Konvergenz führen.

Beispiel fortgeführt: *Werden bei dem Beispielproblem die* $\mu = 4$ *schlechtesten Individuen aus der Population entfernt, so werden als Überlebende für die nächste Generation die Individuen* 00101*,* 11000*,* 01001*,* 01011 *und* 01001 *ausgewählt.*

Die Überlebenden bilden nun die Population der nächsten Generation aus der nun wieder eine Untermenge von Eltern ausgewählt wird. Die Schritte *Selektion der Eltern*, *Kreuzung*, *Mutation* und *Selektion der Überlebenden* werden so oft wiederholt, bis das Abbruchkriterium erfüllt ist. Mögliche Abbruchkriterien sind, dass das Optimum erreicht wurde (falls dieses bekannt ist, wobei dann im Allgemeinen keine Optimierung nötig ist[9]), dass sich die beste Lösung über eine

[9]Ein Ausnahme hierzu bilden Probleme, bei denen es darum geht, eine Lösung zu finden, die verschiedene Randbedingungen befriedigt, z.B. die Erstellung von Stundenplänen für

festgelegte Anzahl an Generationen nicht geändert hat, oder, dass eine festgelegte Anzahl an Generationen durchlaufen wurde.

Es sei hier auch noch erwähnt, dass neben der Wahl geeigneter Parameter und Operatoren für Kreuzung, Mutation und Selektion auch die Populationsgröße ψ einen Einfluss auf die Qualität eines Genetischen Algorithmus hat: Eine zu kleine Population kann zu einer raschen Übernahme der Gesamtpopulation durch ein einziges, relativ gutes Individuum und damit zu einer verfrühten Konvergenz auf ein lokales Optimum führen. Eine zu große Population führt dagegen zu einem erhöhten Rechenaufwand. Zudem bietet sie, insbesondere bei einer sehr stark gewellten Fitnesslandschaft, das Risiko, dass häufig zwei Individuen, die jeweils in der Nähe zweier verschiedener (lokaler) Optima liegen, miteinander gekreuzt werden, und dabei mittelmäßige Nachkommen entstehen, die von beiden Optima weit entfernt sind (Bartlett, 1995)[10].

5.2.3. Behandlung ungültiger Lösungen

Eine wichtige Fragestellung bei der Anwendung von Genetischen Algorithmen auf Probleme mit Randbedingungen ist, wie mit ungültigen Lösungen umgegangen werden soll. Einige Möglichkeiten dazu werden im Folgenden behandelt.

5.2.3.1. Todesstrafe

Die einfachste Möglichkeit zur Behandlung ungültiger Lösungen ist es, diese aus der Population zu entfernen, sie also sozusagen zu „töten“. Praktisch bedeutet dies, dass diese Individuen einen extrem ungünstigen Fitnesswert (z.B. 0 bei einem Maximierungsproblem mit nur positiven Werten) zugewiesen bekommen, der sie von vornherein von der Selektion ausschließt.

Der größte Vorteil dieser Methode ist, dass sie sehr leicht anzuwenden ist. Weitere Berechnungen zur Fitness oder zum Grad der Randbedingungsverletzungen können entfallen.

Dieser Vorteil wird jedoch von verschiedenen Nachteilen überwogen. So führt es zu Stagnation, wenn in der Startpopulation keine gültige Lösung vorhanden ist, da alle Individuen dann die gleiche Fitness haben (Coello Coello, 2002). Michalewicz (1995a) und Coit *et al.* (1996) zeigen zudem auf mehreren Testproblemen, dass die „Todesstrafe“ selbst mit Voraussetzung einer Startpopulation nur aus gültigen Lösungen im Vergleich zu anderen Strategien zur Behandlung ungültiger Lösungen ein wesentlich schlechteres Verhalten zeigte.

Schulen oder Universitäten. Dabei ist bekannt, dass das Optimum bei 0 Randbedingungsverletzungen liegt.

[10] Zur Thematik der Fitnesslandschaften siehe auch Bode (2011)

5.2.3.2. Strafen

Eine weitere Strategie zur Behandlung ungültiger Lösungen ist, diese mit einer Strafe (engl. *penalty*) zu versehen. Die Fitnessfunktion entspricht dann nicht mehr der Zielfunktion $f(x)$ sondern wird ersetzt durch die Funktion $\hat{f}(x)$. Im allgemeinen folgt diese Bestrafung (für ein Minimierungsproblem) der Gleichung

$$\hat{f}(x) = f(x) + \sum_{i=1}^{m} (w_i * d_i^{\kappa}(x)). \tag{5.3}$$

m ist dabei die Anzahl an Randbedingungen, w_i ein Wichtungsfaktor für die Randbedingung i und κ eine Konstante (normalerweise 1 oder 2). $d_i(x)$ ist eine Funktion, die den Abstand der Lösung x vom gültigen Bereich beschreibt. Dies kann bei der Optimierung einer kontinuierlichen Funktion der tatsächliche euklidische Abstand zwischen der Lösung x und der Kurve, die die Randbedingung beschreibt, sein. Bei kombinatorischen Problemen kann dieser Abstand entweder einfach über eine ja/nein-Abfrage, ob die Randbedingung verletzt wurde, oder aber durch die „Kosten“, beispielsweise die Anzahl an Tauschoperationen, die zur Reparatur dieser Lösung, also um aus der ungültigen Lösung eine gültige zu machen, notwendig wären, beschrieben werden,.

Bleibt der Faktor w über den gesamten Optimierungsprozess konstant, so spricht man von einer *statischen Strafe*. Dies hat den Nachteil, dass die Parameter auf jedes neue Problem erst einmal abgestimmt werden müssen, bzw. zum Setzen der w_i die Problemstruktur genauestens bekannt sein muss.

Zur genauen Ausführung der statischen Strafe gibt es eine Vielzahl verschiedener Ansätze. Beispiele dafür sind die Ansätze von Homaifar *et al.* (1994), die verschiedene Grade von Verletzungen der Randbedingungen definieren und die Strafe entsprechend stufenweise quadratisch erhöhen, von Hoffmeister & Sprave (1996), bei denen die Strafe dem Quadrat der Überschreitung der Randbedingung entspricht, sowie der von von Kuri Morales & Villegas Quezada (1998), bei dem der Fitnesswert einer ungültigen Lösung rein über die Anzahl ihrer Randbedingungsverletzungen bestimmt wird.

Ausgehend von der Idee, dass die Bedeutung ungültiger Lösungen am Anfang des Suchvorgangs höher ist, wenn die Region des Suchraumes, in der die guten Lösungen liegen, erst noch gefunden werden muss, als gegen Ende, wenn sich die Suche auf eine oder wenige Regionen beschränkt, kann in die Berechnung von w die Zahl der Generationen eingehen. Geschieht dies direkt, d.h. w wird mit jeder Generation erhöht, so spricht man von *dynamischen Strafen*. Beispiele hierfür sind die Ansätze von Joines & Houck (1994) und Kazarlis & Petridis (1998). In Anlehnung an das Verfahren des *Simulierten Abkühlens* (Kapitel 2.4.4) lässt sich auch ein Schema entwickeln, bei dem w nicht in jeder Generation geändert wird, sondern nach einem zuvor festgelegten Plan. Man spricht dann von *abkühlenden*

Strafen[11]. Einen Ansatz hierfür beschreiben Michalewicz & Attia (1994). Ein gemeinsamer Nachteil der dynamischen und abkühlenden Strafen ist, dass sie, wie auch die statischen Strafen, eine Reihe von Inputparametern benötigen, die jeweils problemabhängig gesetzt werden müssen. Eine falsche Wahl dieser Parameter führt entweder zu vorzeitiger Konvergenz zu einem lokalen Optimum (wenn die Strafen zu hoch sind) oder zu einer Konvergenz zu einer ungültigen Lösung (wenn sie zu niedrig sind).

Um dieses Problem zu umgehen, empfehlen unter anderen Eiben & Smith (2003) die Anpassung der Strafe an den jeweiligen Status der Suche. Dies kann als *adaptive Strafe* geschehen, bei der jeweils die letzten Generationen beobachtet werden. Die Strafen werden dann angepasst, je nachdem ob in den letzten Generationen gute gültige Lösungen gefunden wurden oder nicht. Beispiele hiefür sind die Ansätze von Bean & ben Hadj-Alouane (1992), Coit & Smith (1996) bzw. Coit *et al.* (1996) oder Eiben & Ruttkay (1996).

Eine andere Möglichkeit ist, auch die Strafen einem Evolutionsprozess zu unterwerfen, sogenannte *ko-evolutionäre Strafen*. Ein Ansatz hierzu stammt von Coello Coello (2000b). Hierbei existiert eine zweite Population $P2$, deren Individuen die Straffaktoren beschreiben. Für jedes Individuum aus $P2$ wird eine eigene Kopie von $P1$ erzeugt, für die dann ein oder mehrere Evolutionsschritte (Kreuzung und Mutation) durchgeführt werden. Die Individuen aus $P2$ bekommen dann die durchschnittliche Fitness der gültigen Lösungen, die in ihrer jeweiligen Instanz von $P1$ existieren, als Fitness zugewiesen. Wenn alle Individuen aus $P2$ auf diese Weise einen Fitnesswert erhalten haben, erfolgt nun ein Evolutionsschritt in dieser Population. Danach beginnt der Prozess von vorne. Ein gravierender Nachteil dieses Ansatzes ist jedoch, dass er eine große Anzahl an Auswertungen der Zielfunktion benötigt. Für Probleme, bei denen diese Auswertung viel Zeit beansprucht, ist er also nicht empfehlenswert, sofern sich diese Auswertungen nicht parallelisieren lassen.

5.2.3.3. Reparaturfunktionen

Eine weitere Möglichkeit der Behandlung ungültiger Lösungen ist, diese zu reparieren, das heißt, sie in gültige Lösungen umzuwandeln. Für die Art der Reparaturfunktion gibt es verschiedene Möglichkeiten. Im Folgenden sind einige beispielhafte Anwendungen von Reparaturfunktionen aufgeführt.

Xiao *et al.* (1996) bzw. Xiao *et al.* (1997) beschreiben einen Algorithmus zur Wegsuche für einen Roboter, der sich in einem Raum mit Hindernissen zwischen zwei Orten bewegt. Sie verwenden eine Reparaturfunktion, die Wegpunkte, die innerhalb der Hindernisse liegen aus diesen hinausbewegt und damit den Weg gültig macht. Eine andere Reparaturfunktion, speziell für die Optimierung der Dicke von Laminatplatten, beschreiben Le Riche & Haftka (1995). Da hier die

[11] eigentlich der falsche Begriff, da die Strafen mit der Zeit ja zu- und nicht abnehmen

Dicke der einzelnen Schichten in Kombination mit dem Lastfaktor das Überschreiten der Stabilitätsrandbedingungen beeinflusst, werden diese Größen in der Reparaturfunktion mit einem geeigneten Faktor skaliert. Ein weiteres Beispiel für eine Reparaturfunktion findet sich in Hinterding (1994), hier für das Rucksackproblem: Nach dem Erzeugen eines ungültigen Individuums, das heißt, eines Individuums, das die Gewichtsgrenzen für den Rucksack überschreitet, bleiben nur diejenigen Gegenstände endgültig in dem Individuum enthalten, die beim schrittweisen Hinzufügen keine Verletzung der Randbedingung erzeugen.

Wie diese Beispiele zeigen, ist die Art der Reparaturfunktion sehr stark abhängig vom Problemtyp sowie von der gewählten Repräsentation. Dies bedeutet, sie müssen für jeden Problemtyp neu entworfen werden. Davon abgesehen handelt es sich bei ihnen jedoch um eine sehr gute Möglichkeit zur Behandlung ungültiger Lösungen (Coello Coello, 2002). Allerdings muss bei ihrer Wahl darauf geachtet werden, dass sie den Algorithmus nicht zu stark beeinflussen, indem sie die Suche in eine bestimmte Richtung drängen.

5.2.3.4. Beschränken der Suche auf den gültigen Bereich

Bei einigen Problemen ist es möglich, die Suche rein auf den gültigen Bereich zu beschränken. Anders als bei der Todesstrafe werden hier erst gar keine ungültigen Lösungen erzeugt, indem spezielle Kreuzungs- und Mutationsoperatoren verwendet werden, die nur gültige Lösungen erzeugen können (Kowalczyk, 1997). Alternativ wird eine Kreuzungs- bzw. Mutationsoperation so lange wiederholt, bis eine gültige Lösung gefunden wurde (Eiben & Smith, 2003).

Dieser Ansatz ist allerdings nur auf solche Probleme anwendbar, bei denen der gültige Lösungsraum zusammenhängend ist, da sonst manche gültige Regionen gar nicht erreicht werden können. Auch bei Problemen, bei denen das Optimum auf dem Rand der gültigen Region liegt, kann es zu Schwierigkeiten kommen, wenn die Operatoren das Innere des gültigen Bereichs implizit bevorzugen. Lässt sich allerdings eine Repräsentation für das Problem finden, für die der gültige Lösungsraum geschlossen ist, so lässt sich auf diese Weise viel Rechenzeit sparen. Beispiele für solche Algorithmen finden sich z.B. in Eiben & Smith (2003) für das 8-Damen-Problem und in Chan *et al.* (2001) für die Instandsetzung von Straßenbelag.

5.2.3.5. Sonstige Methoden zur Behandlung ungültiger Lösungen

Neben den in den Abschnitten 5.2.3.1 bis 5.2.3.4 beschriebenen Methoden zur Behandlung ungültiger Lösungen gibt es noch eine Vielzahl weiterer Ansätze. Für einen Überblick siehe Michalewicz (1995b) und Coello Coello (2002).

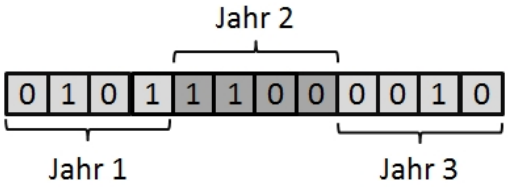

Abbildung 5.3.: Binär-Repräsentation für das Instandsetzungsproblem mit $n = 4$ und $y = 3$. Bauwerk 1 wird im Jahr 2 instandgesetzt, Bauwerk 2 in den Jahren 1 und 2. Eine Beschränkung auf genau l Bauwerke pro Jahr oder eine Kontrolle gegen mehrmaliges Auftauchen desselben Bauwerks im Terminplan ist bei dieser Repräsentation nur schwer zu realisieren.

Erwähnt sei hier noch die Möglichkeit, die Minimierung der Verletzungen der Randbedingungen als zusätzliche Zielfunktion zu definieren und aus dem einkriteriellen Problem somit ein mehrkriterielles zu machen. Eine beispielhafte Anwendung dieser Methode zeigen Surry *et al.* (1995) angewendet auf die Erstellung von Gasleitungsnetzen.

5.3. Genetische Algorithmen für das Lebensdauermanagement

5.3.1. Mögliche Repräsentationen

5.3.1.1. Repräsentationen für ähnliche Probleme aus der Literatur

Die naheliegendste Repräsentation für die Optimierung von Instandsetzungszeitplänen ist sicherlich die Formulierung als Bit-String der Länge $n * y$. Für jedes der n Bauwerke stehen y Einträge zur Verfügung, die entweder den Wert 0 oder den Wert 1 besitzen. Enthält der i-te Eintrag für Bauwerk j eine 1, so wird es in Jahr i instand gesetzt, bei einer 0 nicht (Abbildung 5.3). Eine solche Formulierung verwenden z.B. Liu *et al.* (1997) und Ng *et al.* (2009)[12].

Allerdings lässt sich auf diese Weise nur schwer sicher stellen, dass jedes Bauwerk im Betrachtungszeitraum höchstens einmal und dass pro Jahr eine feste Anzahl von Bauwerken instand gesetzt werden soll. Der Algorithmus produziert auf diese Weise also sehr viele ungültige Lösungen, solange diese Punkte nicht auf andere Weise sicher gestellt werden. Dies wiederum setzt den Entwurf äußerst komplexer Kreuzungs- und Mutationsoperatoren voraus (vgl. Abschnitt 5.2.3.4).

[12] Ng *et al.* (2009) verwenden genau genommen einen Bit-String der Länge $n * y * J$, da sie J verschiedene Instandsetzungsmechanismen betrachten

Einen ähnlichen Ansatz wählen Ferreira *et al.* (2002). Ein Instandsetzungsplan wird dabei durch einen Integerstring der Länge n repräsentiert. Jede Stelle des Strings steht dabei für ein Bauwerk (in diesem Fall einen Straßenabschnitt). Der Wert der Variable an der Stelle j enthält die Information, in welchem Jahr Bauwerk j instand gesetzt werden soll, es gibt also y verschiedene Variablenwerte[13].

Diese Methode stellt sicher, dass jedes Bauwerk im Betrachtungszeitraum genau einmal instand gesetzt wird. Nicht berücksichtigt bleibt aber die Anzahl paralleler Maßnahmen pro Jahr.

5.3.1.2. Für das Lebendauermanagement angepasste Repräsentationen

Zur Vermeidung der oben erwähnten Probleme wurde im Rahmen dieser Arbeit eine andere Repräsentation, die Matrix-Repräsentation, für das Problem entwickelt. Diese folgt intuitiv der Problemstruktur: Eine Lösung wird durch eine Matrix mit y Reihen und l Spalten beschrieben. Einträge in der Matrix sind Ganze Zahlen aus dem Intervall $[1; n]$. Diese stehen für die IDs der einzelnen Bauwerke. Ein Eintrag 17 in der ersten Zeile bedeutet dann, dass das Bauwerk 17 im ersten Jahr des Betrachtungszeitraumes instand gesetzt werden soll. Die Spalte des Eintrags ist ohne weitere Bedeutung (Abbildung 5.4).

Da hier in der Lösungsbeschreibung schon festgelegt ist, dass genau l Bauwerke pro Jahr instand gesetzt werden, muss bei der Erzeugung der Lösungen, für die Startgeneration und später bei Kreuzung und Mutation, nur noch sicher gestellt werden, dass keine Bauwerks-ID mehrfach vorkommt. Dies ist verhältnismäßig einfach möglich. Ungültige Lösungen entstehen damit nur noch, wenn kritische Bauwerke (also Bauwerke, deren Instandsetzungsfrist innerhalb des Betrachtungszeitraumes liegt) zu spät oder gar nicht terminiert werden oder für ein oder mehrere Jahre die Budgetgrenzen über- oder unterschritten werden.

Eine weitere Repräsentation für die Optimierung von Instandsetzungszeitplänen ist denkbar, die Permutations-Repräsentation. Sie wurde entwickelt, um eventuelle Bevorzugungen bestimmter Teile des Lösungsraumes durch die Matrix-Repräsentation zu umgehen. Solche Bevorzugungen können durch eine ungünstige Verknüpfung der genetischen Operatoren entstehen. Genauer ausgeführt wird dies in den Abschnitten 5.3.2.1 und 7.2.1.2. Die Permutations-Repräsentation ist wesentlich weiter abstrahiert vom eigentlichen Problem als die anderen beschriebenen Repräsentationen.

Eine Lösung wird dabei als Permutation der Länge n beschrieben. Die Stelle innerhalb des Chromosoms steht für die Bauwerks-ID (z.B. der erste Eintrag beschreibt das Bauwerk mit der Nummer 1 usw.). In den Einträgen selber ist das Jahr der Instandsetzung folgendermaßen codiert: Die Zahlen 0 bis $l - 1$ stehen

[13] Auch in Ferreira *et al.* (2002) werden J verschiedene Instandsetzungsmaßnahmen betrachtet. Der Variablenwert enthält dann zusätzlich die Information, welche Maßnahme gewählt wird. Es gibt also $y * J$ verscheidene Variablenwerte

27	65	62	88	17	75	82	55	87	37
66	68	53	25	56	67	60	43	6	52
95	74	2	3	14	57	11	36	91	41
70	44	32	13	16	84	24	76	90	5
80	47	85	48	8	83	89	46	50	34

Abbildung 5.4.: Matrix-Repräsentation für das Instandsetzungsproblem mit $n = 100$, $y = 5$ und $l = 10$. Die Zeilen der Matrix entsprechen den Jahren des Terminplanes, die Einträge bezeichnen die Bauwerke über eine ID. Die abgebildete Matrix beschreibt einen Terminplan, in dem im ersten Jahr die Bauwerke mit den Nummern 17, 27, 37, 55, 62, 65, 75, 82, 87 und 88 instand gesetzt werden, im zweiten Jahr die Bauwerke mit den Nummern 6, 25, 43, 52, 53, 56, 60, 66, 67 und 68, usw.

für eine Instandsetzung im ersten Jahr des Betrachtungszeitraumes, die Zahlen l bis $2l - 1$ für das zweite, und so weiter bis zu den Zahlen $(y - 1)l$ bis $yl - 1$ für das letzte Jahr. Einträge größer als $yl - 1$ bedeuten, dass das entsprechende Bauwerk im Betrachtungszeitraum nicht instand gesetzt wird (Abbildung 5.5).

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
90	22	23	58	39	18	69	44	77	73	26	76	33	24	80	34	4	95	67	70	97	85	54	36	13
26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
93	0	96	56	84	89	32	99	49	88	27	9	98	87	71	29	91	17	31	94	47	41	43	63	48
51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
82	19	12	92	7	14	25	50	75	16	78	2	55	51	1	10	15	11	60	30	53	62	86	21	5
76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
37	64	61	79	40	66	6	45	35	42	57	8	3	46	38	28	81	68	74	20	52	59	83	72	65

Abbildung 5.5.: Permutations-Repräsentation für dieselbe Lösung wie in Abbildung 5.4. Das eigentliche Chromosom besteht nur aus den weißen Zellen, die dunkelgrauen Zellen dienen der besseren Orientierung. Bei $y = 5$ und $l = 10$ bedeutet der Eintrag 90 an erster Stelle, dass das Bauwerk 1 im Betrachtungszeitraum von 5 Jahren nicht instand gesetzt wird. Der Eintrag 22 an zweiter Stelle bedeutet eine Instandsetzung für Bauwerk 2 im Jahr 3 ($2l \leq 22 < 3l - 1$).

Bei Anwendung entsprechender auf Permutationen zugeschnittener Operatoren zur Kreuzung und Mutation ist bei dieser Lösungsrepräsentation von vornherein sicher gestellt, dass kein Bauwerk doppelt im Zeitplan vorkommt und pro Jahr genau l Bauwerke instand gesetzt werden. Damit wird die Suche wesentlich freier, da nicht künstlich bestimmte Operationen verhindert werden müssen, was immer die Gefahr einer zu starken Beeinflussung beinhaltet.

5.3.2. Kreuzungs- und Mutationsoperatoren

5.3.2.1. Kreuzung und Mutation für die Matrix-Repräsentation

Für die Zielfunktion, also den Einfluss eines Terminplanes auf den Verkehr, ist es von Bedeutung, welche Bauwerke gemeinsam in welchem Jahr instand gesetzt werden. Diese Information ist bei der Matrix-Repräsentation in den Matrixzeilen enthalten. Ein geeigneter Kreuzungsoperator sollte also im Hinblick auf den Respekt (vgl. Abschnitt 5.2.2.2) möglichst ganze Zeilen aus den beiden Elternchromosomen übernehmen.

Ein Operator, der dieser Anforderung entspricht, wird im Folgenden beschrieben: Zunächst werden $\frac{y}{2}$ ganze Zufallszahlen zwischen 0 und $y-1$ erzeugt[14]. Die Zeilen mit diesen Nummern werden direkt aus dem Chromosom des ersten Elternteils übernommen (Abbildung 5.6). Die restlichen Zeilen werden aus denen des zweiten Elternchromosoms aufgefüllt. Dabei ist jedoch zu beachten, dass Einträge, die bereits im Chromosom enthalten sind, nicht ein zweites Mal übernommen werden dürfen. Tritt also beim Übertragen der Werte aus dem zweiten Elternchromosom ein Wert auf, der bereits in einem der Jahre vorhanden ist, die aus dem ersten Elternteil übernommen wurden, so muss dieses ersetzt werden. Dazu wird ein zufälliges Bauwerk ausgewählt, das bisher noch nicht Teil des Kindchromosoms ist, und an Stelle des sonst doppelt auftretenden Elements geschrieben (Abbildung 5.7). Dieser Operator übernimmt also $\frac{y}{2}$ Jahre direkt aus einem Elternteil, die restlichen mit leichten Modifikationen aus dem zweiten (Abbildung 5.8). Ein zweites Kind lässt sich mit vertauschten Rollen der Eltern erzeugen.

Alternative Kreuzungsoperatoren wären Ein-Punkt- oder n-Punkt-Kreuzung, wobei die Elternchromosomen an einem zufällig gewählten Punkt (bzw. n zufällig gewählten Punkten) geteilt werden. Jeweils die Hälfte der so entstandenen Chromosomenteilstücke werden nun von dem einen Elternchromosom, die restlichen (natürlich mit der oben erwähnten Überprüfung auf doppelte Elemente) von dem anderen Elternchromosom in das Kindchromosom übernommen. Ein Nachteil hierbei ist, dass das Kind dabei weniger zusammenhängende Jahre des Terminplanes unverändert von den Eltern übernimmt.

[14] Bei ungeradem y kann entweder auf- oder abgerundet werden. Da das zweite Kind entsprechend mit vertauschten Eltern erzeugt wird, bedeutet dies keinen Unterschied.

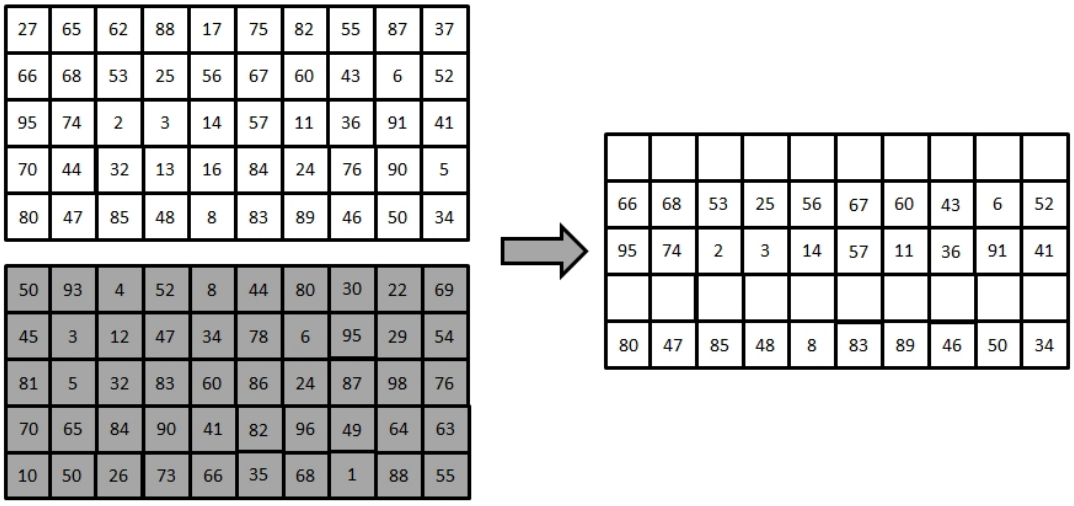

Abbildung 5.6.: Erster Schritt der Kreuzung mit Genen der Matrix-Repräsentation: $\frac{y}{2}$ zufällig gewählte Zeilen werden direkt aus dem ersten Elternchromosom (hier weiß dargestellt) übernommen.

Der Mutationsoperator muss bei dieser Repräsentation dafür sorgen, dass bisher nicht berücksichtigte Bauwerke Teil des Terminplanes werden, so dass es möglich ist, alle Bereiche des Lösungsraumes zu erreichen. Ein geeigneter Operator hierfür verhält sich ähnlich wie der für die Kreuzung beschriebene Operator zur Kontrolle, dass keine Doppeleinträge im Chromosom entstehen: Der Eintrag des zu mutierenden Gens wird gegen ein zufällig gewähltes, bisher nicht im Terminplan vertretenes Bauwerk ausgetauscht.

In Kombination mit dem jahresweisen Kreuzungsoperator sowie der in Abschnitt 5.3.3 beschriebenen Reparaturfunktion kann dieser Mutationoperator jedoch dazu führen, dass die Suche sich in einem lokalen Optimum verfängt: Durch den Kreuzungsoperator werden nur ganze Jahrespläne ausgetauscht. Da der Mutationsoperator nur einen Tausch mit nicht berücksichtigten Bauwerken durchführt, kann ein Jahreswechsel eines kritischen Bauwerks nur über den Umweg einer ungültigen Lösung (in der dieses Bauwerk nicht berücksichtigt wird)[15] und anschließender Reparatur stattfinden. Vor allem in späteren Generationen, wenn die Individuen nur noch sehr wenig bis gar keine Randbedingungsverletzungen aufweisen, fügt die Reparaturfunktion das kritische Bauwerk häufig wieder genau in das Jahr ein, aus dem es entnommen wurde, da an seinem alten Platz mit hoher Wahrscheinlichkeit ein unkritisches Bauwerk steht (da die Lösung vor der Mutation wahrscheinlich gültig war).[16] Eine Ein-Punkt- bzw. n-Punkt-Kreuzung

[15] Eine soche Lösung kann nur durch Mutation entstehen, wenn ein kritisches Bauwerk durch ein bisher nicht berücksichtigtes Bauwerk ersetzt wird.

[16] Genau genommen existiert noch eine zweite Möglichkeit für einen Wechsel eines kritischen Bauwerks zwischen zwei Jahren: Wenn bei der Kreuzung beim Übertragen der Einträge aus dem zweiten Elternteil ein Doppeleintrag entstehen würde, wird statt dessen ein zufälliges

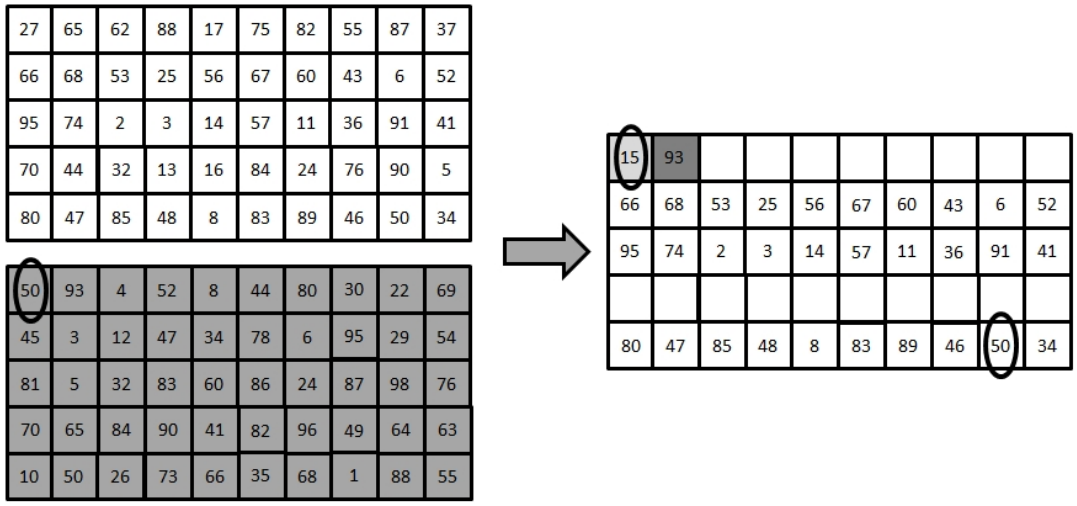

Abbildung 5.7.: Fortsetzung der Kreuzung aus Abbildung 5.6: Die verbliebenen Zeilen des Kindchromosoms werden aus dem zweiten Elternchromosom (dunkelgrau) aufgefüllt. Der Eintrag 50 an der ersten Stelle des zweiten Elternchromosoms kann nicht übernommen werden, da das Kindchromosom schon eine 50 an der neunten Stelle der fünften Zeile besitzt. Stattdessen wird eine zufällige, noch nicht im Kind vorkommende Zahl, hier 15, an die entsprechende Stelle geschrieben. Die 93 an zweiter Stelle kann direkt aus dem zweiten Elternchromosom entnommen werden.

zeigt dieses Verhalten nicht, da hier (fast) immer ein oder mehrere Jahre des Terminplanes aufgebrochen werden.

Aus diesen Gründen muss bei Verwendung des jahresweisen Kreuzungsoperators ein zusätzlicher Mutationsoperator eingeführt werden, der den Austausch von Einträgen zwischen den einzelnen Matrixzeilen ermöglicht. Dabei tauscht ein Eintrag mit einem zufällig gewählten Eintrag einer anderen Zeile den Platz.

Eine Änderung sollte im Schnitt einmal pro Chromosom geschehen, damit sich die neue Lösung nicht zu weit von ihrer Ausgangslösung entfernt. Daher sollte $p_{mutation} = \frac{1}{yl}$ gewählt werden. Bei Verwendung der jahresweisen Kreuzung wird zwischen den beiden verschiedenen Mutationsoperatoren gewechselt: Mit einer Wahrscheinlichkeit von 50% handelt es sich bei der Mutation um einen Tausch mit bisher nicht berücksichtigten Bauwerken, sonst um einen Tausch mit einem Eintrag aus einer anderen Zeile der Chromosommatrix. Bei der Verwendung der Ein-Punkt- bzw. n-Punkt-Kreuzung kann der Tausch immer mit den bisher nicht berücksichtigten Bauwerken erfolgen.

Bauwerk gewählt, das zu diesem Zeitpunkt noch nicht im Kind vorhanden ist. Dabei kann es sich auch um ein Bauwerk handeln, das in zweiten Elternchromosom in einem der aus dem ersten Elternchromosom übernommenen Jahre (dieser Fall trat z.B. für das Bauwerk 10 in der Kreuzung in Abbildung 5.8 ein) oder aber in einem bisher noch gar nicht in das Kind übertragenem Jahr vorkommt, handeln. Die Wahrscheinlichkeit hierfür ist jedoch sehr gering.

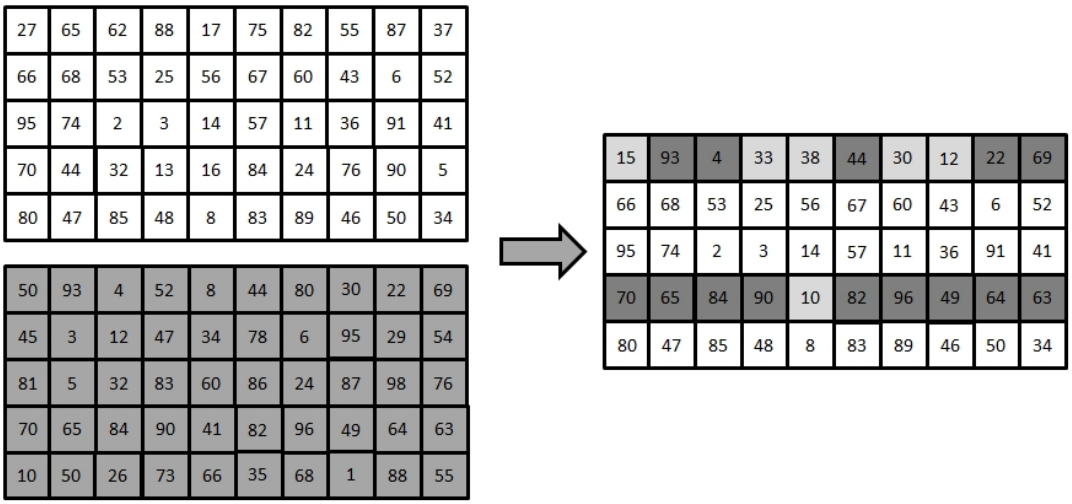

Abbildung 5.8.: Der Abschluss der Kreuzungsoperation aus Abbildungen 5.6 und 5.7: Drei Jahre wurden direkt aus dem ersten Elternchromosom übernommen (weiß). Die restlichen Jahre konnten nur zum Teil aus dem zweiten Elternchromosom aufgefüllt werden (dunkelgrau). Wo dies nicht möglich war, da sonst Doppeleinträge entstanden wären, wurden zufällig gewählte gültige Einträge eingefügt (hellgrau).

5.3.2.2. Kreuzung und Mutation für die Permutations-Repräsentation

In der Permutations-Repräsentation wirkt es sich auf die Zielfunktion aus, an welcher Stelle genau eine Zahl der Permutation steht, nicht aber die relative Reihenfolge der Zahlen. Ein geeigneter Kreuzungsoperator bewahrt also erstere Eigenschaft der Elternchromosomen.

Ein Operator, der diese Voraussetzung weitestgehend erfüllt, ist das *Partially Mapped Crossover* (Goldberg & Lingle Jr, 1985). Dieses folgt den Schritten:

1. Zwei zufällige Kreuzungspunkte werden gewählt und der Abschnitt dazwischen aus dem Chromosom des ersten Elternteils in das Kindchromosom kopiert (Abbildung 5.9, oben).

2. Beginnend vom ersten Kreuzungspunkt wird im Chromosom des zweiten Elternteils nach Elementen gesucht, die noch nicht Teil des Kindchromosoms sind. Wird ein solches Element i gefunden, so wird überprüft, welches Element j im Kindchromosom an dessen Stelle steht. Dieses Element i wird an die Stelle im Kindchromosom geschrieben, die j im Chromosom des zweiten Elternteils einnimmt. Ist diese Stelle bereits durch ein Element k besetzt, so wird i an die Stelle, die k im zweiten Elternchromosom hat, geschrieben, usw. (Abbildung 5.9, Mitte).

3. Die restlichen freien Stellen des Kindchromosoms werden durch direkte Kopie aus dem zweiten Elternchromosom aufgefüllt (Abbildung 5.9, unten).

Die einzelnen Schritte seien noch einmal an dem Beispiel aus Abbildung 5.9 erläutert:

Zunächst werden als Kreuzungspunkte die 4. und die 7. Stelle zufällig gewählt. Die Elemente dazwischen werden aus dem ersten Elternchromosom in das Kindchromosom kopiert.

Im zweiten Elternchromosom steht an der vierten Stelle eine 8 ($i = 8$). Diese 8 ist bisher noch nicht Teil des Kindchromosoms. An der vierten Stelle steht dort eine 4 ($j = 4$). Im zweiten Elternchromosom steht die 4 an der neunten Stelle. Daher wird die 8 im Kindchromosom an die neunte Stelle geschrieben.

Auch die 2 auf der fünften Stelle des zweiten Elternchromosoms ist noch nicht Teil des Kindchromosoms ($i = 2$). An der entsprechenden Stelle dort steht eine 5 ($j = 5$). Diese steht im zweiten Elternchromosom an der siebten Stelle, welche im Kindchromosom bereits durch eine 7 belegt ist ($k = 7$). Die 2 kommt im Kindchromosom also an die Stelle, die die 7 im zweiten Elternchromosom hat, also an die dritte Stelle.

Die 6 und 5 an sechster und siebter Stelle des zweiten Elternchromosoms sind bereits Teil des Kindchromosoms und müssen nicht mehr beachtet werden. Die jetzt noch leeren Stellen des Kindchromosoms werden aus dem zweiten Elternchromosom befüllt.

Wie der unter 5.3.2.1 beschriebene Kreuzungsoperator für die Matrix-Repräsentation übernimmt auch dieser Operator die Informationen aus dem ersten Elternteil direkt, die Informationen aus dem zweiten Elternteil müssen dagegen teilweise angepasst werden. Auch hier lässt sich ein zweites Kind durch Vertauschen der Elternrollen erzeugen.

Der gewählte Mutationsoperator für die Permutations-Repräsentation ist sehr einfach: Zwei Einträge des Chromosoms werden zufällig ausgewählt und miteinander vertauscht. Im Grunde genommen bedeutet dies, dass die Instandsetzungsjahre der beiden entsprechenden Bauwerke miteinander vertauscht werden. Da dieser Operator das gesamte Chromosom betrifft, bezieht sich die Wahrscheinlichkeit p_{mutation} hier darauf, ob er überhaupt auf das betrachtete Individuum angewandt wird.

5.3.3. Behandlung ungültiger Lösungen

Zu unterscheiden sind zwei Typen der Randbedingungsverletzung, eine Verletzung der Sicherheitsrandbedingung oder eine der Kostenrandbedingung. Die Wahrscheinlichkeit ersterer lässt sich leicht bestimmen: Es ist bekannt, wie viele Bauwerke kritisch sind, d.h. ihre Instandsetzungsfrist innerhalb des Betrachtungszeitraumes haben. Zudem ist bekannt, welcher Anteil aller betrachteten Bauwerke nicht in den Terminplan aufgenommen wird. Aus beiden Größen lässt sich be-

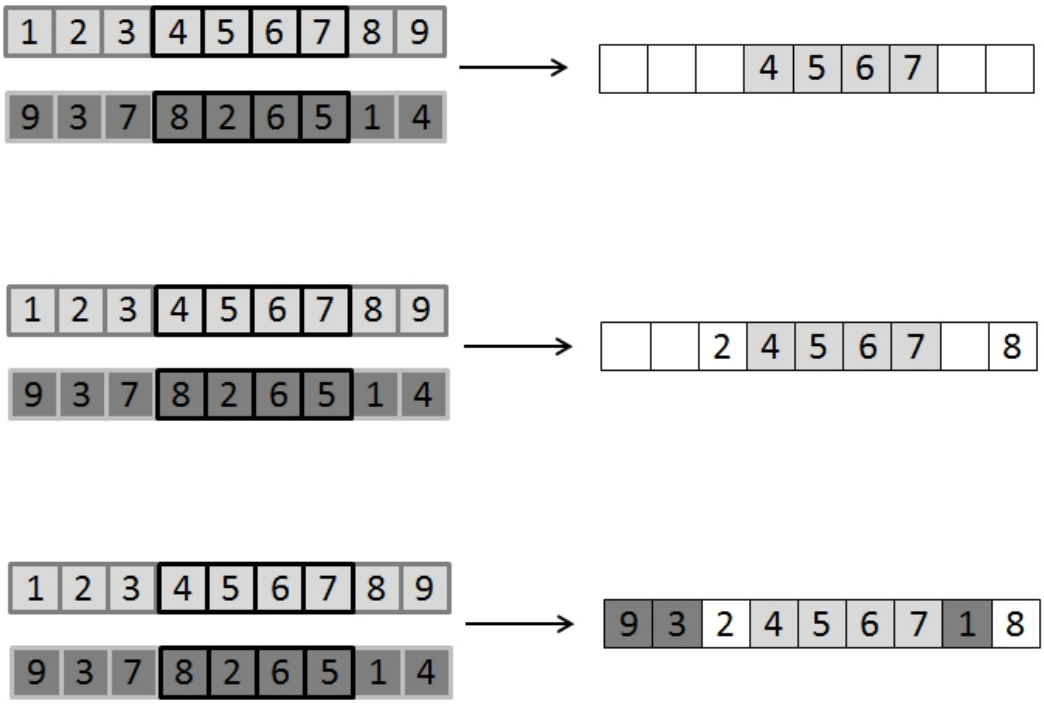

Abbildung 5.9.: *Partially Mapped Crossover*: Im ersten Schritt wird der Abschnitt zwischen zwei zufällig gewählten Kreuzungspunkten aus dem ersten Elternchtomosom unverändert in das Kindchromosom kopiert. Beginnend vom ersten Kreuzungspunkt wird im Chromosom des zweiten Elternteils dann nach Elementen gesucht, die noch nicht Teil des Kindchromosoms sind. Diese werden dann an die Stelle des Kindchromosoms kopiert, die das entsprechende Element aus dem Kindchromosom (bzw. ersten Elternchromosom) im zweiten Elternchromosom einnimmt. Das restliche Kindchromosom wird aus dem zweiten Elternchromosom aufgefüllt. (Abbildung nach Eiben & Smith (2003))

stimmen, wie viele kritische Bauwerke voraussichtlich nicht Teil des Terminplanes werden, also vereinfacht wie viele Verletzungen der Sicherheitsrandbedingung in einem zufällig erstellten Terminplan voraussichtlich entstehen[17].

Anders ist es mit der Kostenrandbedingung. Ihr voraussichtlichr Verletzungsgrad ist nicht so leicht zu ermitteln. Aus den beiden Grenzen der Kostenrandbedingung und den Reparaturkosten der einzelnen Bauwerke lässt sich jedoch eine grobe Abschätzung treffen, für wie viele Jahre eines zufällig erstellten Zeitplanes diese Randbedingung verletzt wird.

[17] Da für alle kritischen Bauwerke auch die tatsächliche Frist bekannt ist, lässt sich die Größe auch genauer bestimmen. Der vereinfachte Ansatz liefert jedoch schon gute Ergebnisse.

5.3.3.1. Struktur des Lösungsraumes

Beide gewählten Repräsentationen für die Optimierung der Instandsetzungszeitpläne haben den Nachteil, dass in ihnen die gültigen Bereiche des Lösungsraumes nicht zusammenhängend sind. Das bedeutet, dass von einer gültigen Lösung aus rein über Nachbarschaftsbeziehungen nicht alle anderen gültigen Lösungen erreicht werden können, wenn nicht zwischendurch auch ungültige Lösungen zugelassen werden. Zwei Lösungen sind benachbart, wenn die eine durch Mutation eines einzelnen Gens aus der anderen erstellt werden kann (vgl. Abbildung 5.10).

Bei der Matrix-Repräsentation kann eine gültige Lösung in eine ungültige umgewandelt werden, indem ein kritisches Bauwerk durch ein unkritisches ersetzt wird. Aus dieser ungültigen Lösung lässt sich wieder eine gültige Lösung erstellen, indem das erwähnte kritische Bauwerk, durch Tausch mit einem unkritischen Bauwerk, wieder in den Terminplan aufgenommen wird. Wird es dabei in ein anderes Jahr geschrieben, als das, in dem es zuvor stand, so ist diese neue gültige Lösung von der anderen nur über den Zwischenschritt über die ungültige Lösung zu erreichen[18]. Für die Permutations-Repräsentation gelten diese Überlegungen entsprechend.

Die gültigen Lösungen bilden zudem einen sehr geringen Anteil des gesamten Lösungsraumes. Zur Abschätzung, welchen Anteil des Lösungsraumes die gültigen Lösungen einnehmen, seien hier die beiden Extremfälle unter reiner Berücksichtigung der Sicherheitsrandbedingung betrachtet: In dem Fall, dass die Anzahl der Bauwerke, deren Frist pro betrachtetem Jahr abläuft, gleich der Anzahl der pro Jahr instand zu setzenden Bauwerke l ist, gibt es genau eine gültige Lösung. Der andere Extremfall tritt ein, wenn pro betrachtetem Jahr genau für ein Bauwerk die Frist abläuft. Die restlichen Stellen des Terminplans können dann frei mit unkritischen Bauwerken befüllt werden. Für ein Beispiel mit $n = 100$, $y = 5$ und $l = 10$ ergibt sich damit eine Gesamtmenge von $4.98 * 10^{57}$ gültigen Lösungen. Bei einer Lösungsraumgröße von $4.88 * 10^{60}$ (vgl. Abschnitt 3.3) bedeutet dies, dass nur 0.1% aller Lösungen gültig sind. Da normalerweise mehr kritische Bauwerke vorhanden sind und die Kostenrandbedingung den gültigen Bereich zusätzlich einschränkt, ist davon auszugehen, dass dieser Anteil in der praktischen Anwendung noch deutlich kleiner ist.

Daher ist es sehr unwahrscheinlich in einer zufällig erstellten Startpopulation überhaupt gültige Lösungen zu finden. In Testläufen (vgl. Kapitel 7) zeigte sich sogar, dass die Lösungen der Startpopulationen zumeist nicht nur ungültig sind, sondern auch einen sehr hohen Grad an Randbedingungsverletzungen aufweisen.

[18] Mit dem in Abschnitt 5.3.2.1 erwähnten zweiten Mutationsoperator befindet sich die neue gültige Lösung direkt in der Nachbarschaft der ersten Lösung. Durch die Kostenrandbedingung entstehen aber auch hier unzusammenhängende Bereiche, beispielsweise, wenn die Reparaturkosten für das verschobene kritische Bauwerk sehr hoch sind und aus dem Jahr, in dem es in der neuen Lösung zu stehen kommt erst noch ein anderes teures Bauwerk entfernt werden muss, damit die Lösung gültig ist.

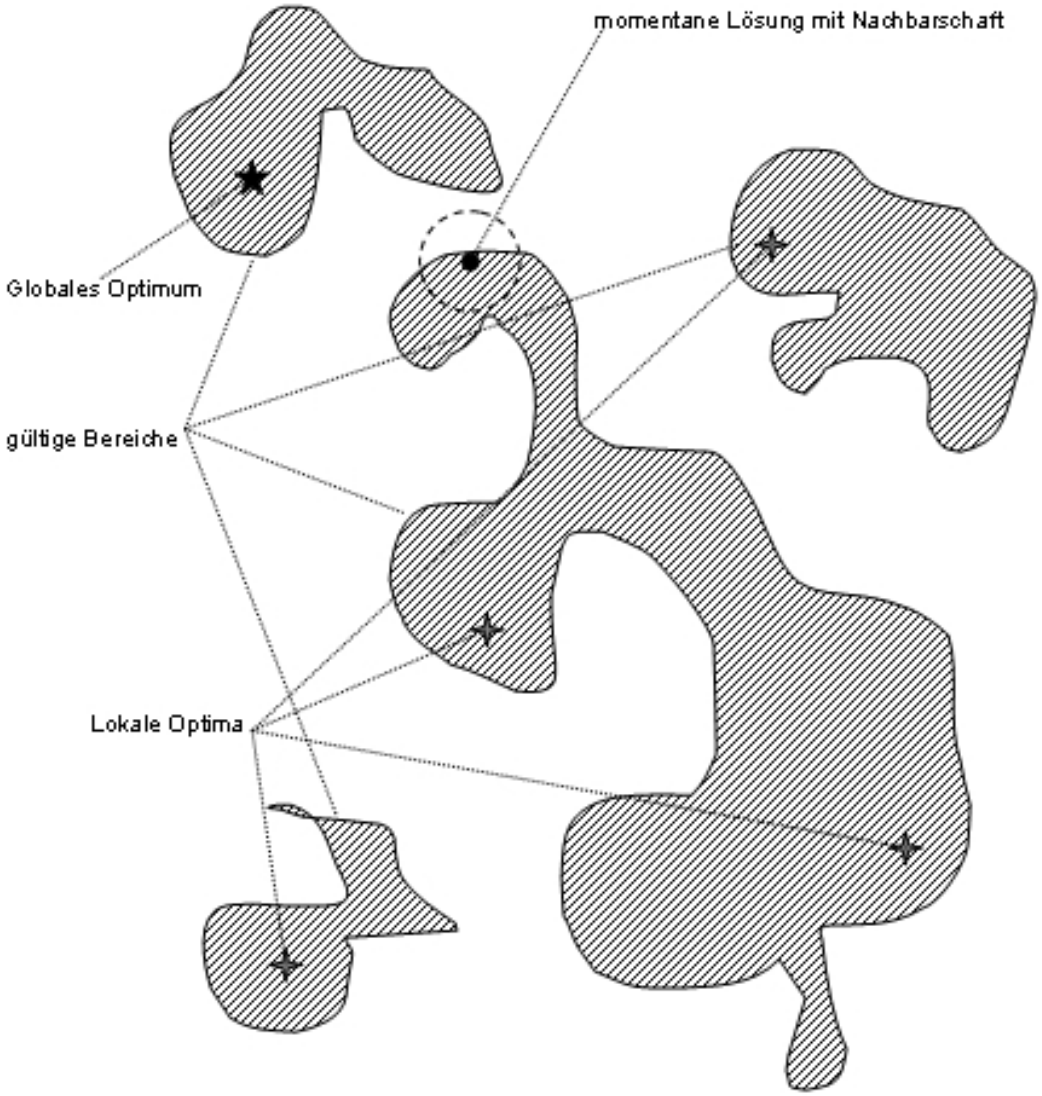

Abbildung 5.10.: Schematische Darstellung der gültigen Bereiche innerhalb des Lösungsraumes. Sie sind stark zerfasert und hängen nicht zusammen. Um von der als schwarzer Kreis dargestellten momentanen Lösung zu dem Bereich zu kommen, der das globale Optimum enthält, müssen ungültige Lösungen als Zwischenlösungen zugelassen werden, da die Nachbarschaft nicht groß genug ist, den Spalt zwischen beiden gültigen Bereichen zu überbrücken. Sonst ist die Suche in den Bereichen gefangen, in denen per Zufall die ersten gültigen Lösungen gefunden wurden.

Aus diesen Gründen ist es erforderlich, geeignete Maßnahmen zur Behandlung der ungültigen Lösungen einzuführen. Von den in Abschnitt 5.2.3 beschriebenen Methoden kommen nur Strafen (5.2.3.2) und Reparaturfunktionen (5.2.3.3) in Frage: Eine Todesstrafe (5.2.3.1) fällt aus, da in der Startpopulation keine gültigen Lösungen zu erwarten sind. Ähnliches gilt für die Beschränkung auf den gültigen Bereich (5.2.3.4): Schon das Erzeugen von gültigen Lösungen stellt sich als Schwierigkeit heraus.

5.3.3.2. Strafen

Als einfachster Ansatz werden statische Strafen (siehe Abschnitt 5.2.3.2) gewählt. Dies ist daher möglich, da der erwartete Grad der Verletzung sowie die erwartete Größenordnung für die Zielfunktion recht gut bekannt sind:

Es hat sich als vorteilhaft erwiesen, die Strafen so zu wählen, dass der Fitnesswert für eine Lösung mit durchschnittlich vielen Randbedingungsverletzungen in etwa dem doppelten Zielfunktionswert entspricht. Dazu wird für eine neue Probleminstanz der Zielfunktionswert für eine zufällig generierte Lösung ermittelt. Dieser wird durch die erwartete Anzahl an Randbedingungsverletzungen geteilt. Der so entstandene Faktor w wird, multipliziert mit der Menge der tatsächlich aufgetretenen Randbedingungsverletzungen g, auf den Zielfunktionswert $f(x)$ jeder ungültigen Lösung addiert. Der Fitnesswert einer Lösung x berechnet sich also aus

$$\hat{f}(x) = f(x) + w * g(x) \tag{5.4}$$

Der Wert w lässt sich auch als Ausgangspunkt für eine adaptive Strafe nutzen, zum Beispiel, indem w erhöht wird, falls in den letzten Generationen keine neuen gültigen Lösungen erzeugt wurden, und gesenkt wird, wenn die Suche stagniert.

5.3.3.3. Minimierung des Verletzungsgrades

Da in den ersten Generationen des Algorithmus bei beiden Repräsentationen mit sehr hoher Wahrscheinlichkeit keine gültigen Lösungen vorhanden sind, liegt der Schwerpunkt hier auf der Suche nach gültigen Lösungen. Über die Bestrafung nach Gleichung 5.4 werden Lösungen mit geringerem Verletzungsgrad zwar bevorzugt, allerdings hat ihre Verwendung in diesen frühen Generationen zwei Nachteile:

Zum einen verlangt sie eine Auswertung der Zielfunktion (mit dem Verkehrssimulator) für jede erzeugte ungültige Lösung. Da es mehrere Generationen dauern kann, bis überhaupt eine gültige Lösung gefunden wird, handelt es sich hier um sinnlos verbrauchte Rechenzeit.

Außerdem kann es sein, dass eine Lösung mit sehr vielen Randbedingungsverletzungen aber einem außerordentlich guten Zielfunktionswert über die Strafe

nach Gleichung 5.4 einen besseren Fitnesswert erhält als eine andere Lösung mit niedrigem Verletzungsgrad aber einem schlechteren Zielfunktionswert. In späteren Generationen, wenn schon gültige Lösungen in der Population vorhanden sind, ist dieses Verhalten nicht schädlich und kann sogar nützlich sein, um die Suche von einer gültigen Region auf eine weitere auszuweiten. In den ersten Generationen, in denen es nur um die Suche nach einem gültigen Bereich geht, kann es jedoch dazu führen, dass sich die Suche wieder davon entfernt.

Aus diesen Gründen bietet es sich an, die Fitnessfunktion in diesen frühen Generationen durch die Minimierung der Anzahl der Randbedingungsverletzungen zu ersetzen: So lange keine gültige Lösung bekannt ist, ist die Fitness eines Individuums x also gleich $g(x)$. Sobald das erste gültige Individuum gefunden wurde, wird die Fitnessfunktion für alle zu diesem Zeitpunkt existierenden Individuen (das sind diejenigen aus der Elternpopulation, sowie alle in dieser Generation vor der gültigen Lösung erzeugten Individuuen) dann wieder durch die aus Gleichung 5.4 ersetzt (das heißt für alle diese Individuen muss nun eine Auswertung mit dem Verkehrssimulator erfolgen). Auch alle weiteren Individuen, die in dieser Generation sowie allen nachfolgenden Generationen erzeugt werden, bestimmen nun ihre Fitness nach Gleichung 5.4.

5.3.3.4. Reparaturfunktionen

Zusätzlich zu den in den Abschnitten 5.3.3.2 und 5.3.3.3 beschriebenen Maßnahmen lassen sich auch noch Reparaturfunktionen einsetzen. Diese können und müssen in Kombination mit den anderen Maßnahmen eingesetzt werden, da es sein kann, dass einzelne Lösungen sich nicht vollständig reparieren lassen, das heißt, dass ein Teil der Randbedingunngsverletzungen auch nach der Reparatur erhalten bleibt. Die Anwendung der Reparaturfunktionen führt aber dazu, dass schneller, das heißt in früheren Generationen, gültige Lösungen gefunden werden. Auch ist die Wahrscheinlichkeit höher, dass die Suche relativ zeitgleich verschiedene gültige Regionen erreicht und damit die Diversität unter den gültigen Lösungen erhöht wird.

Reparaturfunktion für Matrix-Repräsentation Eine Lösung verletzt dann die Sicherheitsrandbedingung, wenn ein kritisches Bauwerk, also ein Bauwerk mit Instandsetzungsfrist innerhalb des Betrachtungszeitraumes, nicht oder zu spät im Zeitplan berücksichtigt wurde. Für die Matrix-Repräsentation bedeutet dies, ein Bauwerk mit Instandsetzungsfrist im Jahr j befindet sich in Zeile $j+a$ ($a > 0$) oder taucht innerhalb des Chromosoms überhaupt nicht auf.

Zur Reparatur wird die Liste der kritischen Bauwerke Schritt für Schritt durchlaufen und für jedes überprüft, ob es vor Ablauf seiner Instandsetzungsfrist im Terminplan aufgenommen wurde. Die Reihenfolge, in der diese Überprüfung abläuft, sollte zufällig sein, da sonst bestimmte Regionen des Lösungsraumes durch die

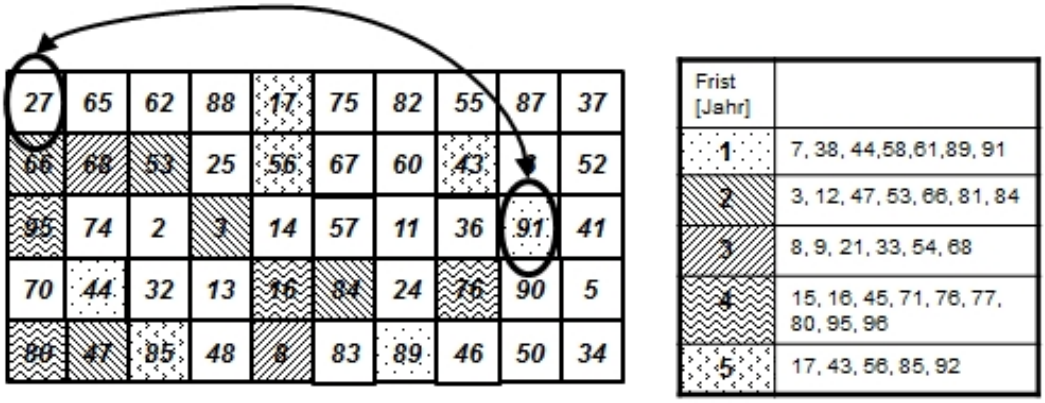

Abbildung 5.11.: Reparatur einer ungültigen Lösung in der Matrix-Repräsentation: Das kritische Bauwerk 91 mit Instandsetzungsfrist im Jahr 1 steht im Jahr 3 des Terminplanes. Im Jahr 1 gibt es ein unkritisches Bauwerk 27. Mit diesem tauscht Bauwerk 91 seinen Platz im Terminplan.

Reparaturfunktion bevorzugt werden. Als Ergebnis der Überprüfung gibt es fünf Fälle:

- Das Bauwerk steht vor Ablauf seiner Instandsetzungsfrist im Terminplan: Hier sind keine weiteren Maßnahmen nötig.
- Das Bauwerk steht im Terminplan aber nach Ablauf seiner Instandsetzungsfrist; in den Jahren vor Ablauf der Instandsetzungfrist gibt es mindestens ein unkritisches Bauwerk im Terminplan: Das kritische Bauwerk tauscht den Platz mit einem der unkritischen Bauwerke aus einem der für es in Frage kommenden Jahre (Abbildung 5.11).
- Das Bauwerk steht im Terminplan aber nach Ablauf seiner Instandsetzungsfrist; in den Jahren vor Ablauf der Instandsetzungfrist gibt es kein unkritisches Bauwerk im Terminplan: Innerhalb der für das kritische Bauwerk in Frage kommenden Jahre wird nach dem Bauwerk mit der höchsten Instandsetzungsfrist gesucht. Nun wird für dieses Bauwerk ein unkritisches Bauwerk innerhalb seiner Instandsetzungsfrist gesucht. Die drei Bauwerke werden ringweise vertauscht (Abbildung 5.12).
- Das Bauwerk steht nicht im Terminplan; in den Jahren vor Ablauf der Instandsetzungfrist gibt es mindestens ein unkritisches Bauwerk im Terminplan: Das kritische Bauwerk wird an die Stelle des unkritischen geschrieben. Das unkritische Bauwerk ist danach nicht mehr Teil des Terminplanes (Abbildung 5.13).
- Das Bauwerk steht nicht im Terminplan; in den Jahren vor Ablauf der Instandsetzungfrist gibt es kein unkritisches Bauwerk im Terminplan: In-

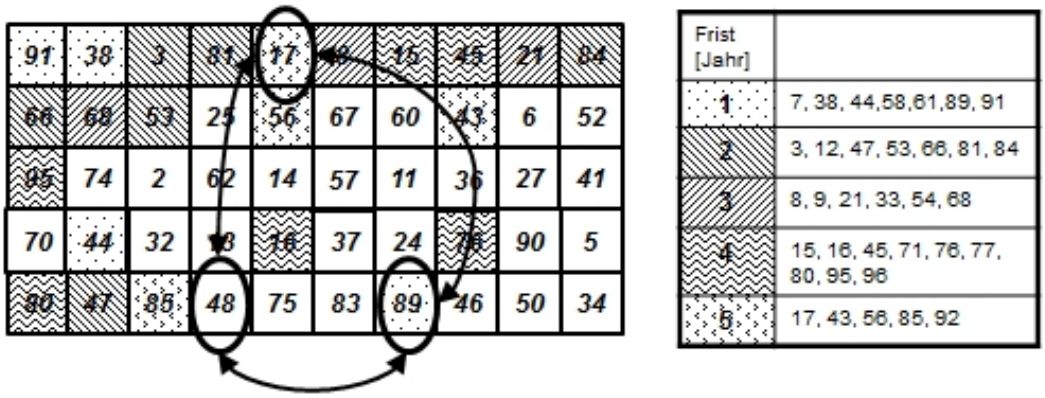

Abbildung 5.12.: Reparatur einer ungültigen Lösung in der Matrix-Repräsentation: Das kritische Bauwerk 89 mit Instandsetzungsfrist im Jahr 1 steht im Jahr 5 des Terminplanes. Im Jahr 1 gibt es kein unkritisches Bauwerk. Das Bauwerk 17 hat von den Bauwerken innerhalb der Frist von Bauwerk 89 die höchste Frist. Ein unkritisches Bauwerk innerhalb der Instandsetzungsfrist von 17 ist das Bauwerk 48. 17 nimmt dessen Platz ein, 89 den alten Platz von 17 und 48 den alten von 89.

nerhalb der für das kritische Bauwerk in Frage kommenden Jahre wird nach dem Bauwerk mit der höchsten Instandsetzungsfrist gesucht. Nun wird für dieses Bauwerk ein unkritisches Bauwerk innerhalb seiner Instandsetzungsfrist gesucht. Das kritische Bauwerk mit der höheren Instandsetzungsfrist wird an die Stelle des unkritischen Bauwerks geschrieben, das kritische Bauwerk mit der niedrigeren Instandsetzungsfrist nimmt seinen Platz ein. Das unkritische Bauwerk ist danach nicht mehr Teil des Terminplanes (Abbildung 5.14).

Reparaturfunktion für Permutations-Repräsentation Für die Permutations-Repräsentation ist die Sicherheitsrandbedingung dann verletzt, wenn Bauwerk i eine Instandsetzungsfrist im Jahr j hat und an der Position i ein Wert größer $lj-1$ steht. Auch hier werden die kritischen Bauwerke nacheinander abgearbeitet. Im Gegensatz zur Matrix-Repräsenatation besteht hier keine Gefahr, dass die Suche in bestimmte Bereiche des Lösungsraumes gedrängt wird, daher können die Bauwerke auch nach ihrer ID geordnet untersucht werden.

Hierbei sind nur zwei Fälle zu unterscheiden:

- An der Position des kritischen Bauwerks mit der Instandsetzungsfrist j steht ein Wert kleiner lj: Hier sind keine weiteren Maßnahmen nötig.
- An der Position des kritischen Bauwerks mit der Instandsetzungsfrist j steht ein Wert größer $lj-1$: Es wird ein unkritisches Bauwerk gesucht, an dessen

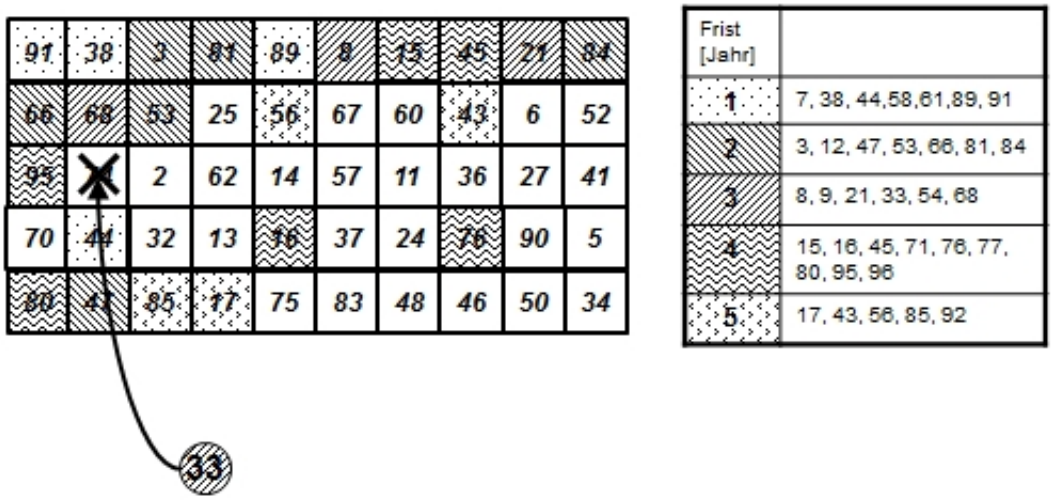

Abbildung 5.13.: Reparatur einer ungültigen Lösung in der Matrix-Repräsentation: Das kritische Bauwerk 33 mit Instandsetzungsfrist im Jahr 3 ist nicht Teil des Terminplanes. Innerhalb seiner Instandsetzungsfrist gibt es das unkritische Bauwerk 74. Dieses wird aus dem Terminplan entfernt und 33 nimmt seinen Platz ein.

Position ein Wert kleiner lj steht. Die Einträge dieser beiden Positionen werden miteinander vertauscht (Abbildung 5.15).

Bei den beschriebenen Reparaturfunktionen beider Repräsentationen kann es auftreten, dass für einzelne kritische Bauwerke kein gültiger Tauschpartner gefunden werden kann. Die betroffenen Individuen bleiben dann auch nach der Reparatur ungültig. Ebenso können Verletzungen der Kostenrandbedingung auf diese Weise nicht behandelt werden, schlimmstenfalls kommt es bei der Reparatur sogar zu einer zusätzlichen Verletzung von ihr. Im Allgemeinen geht die Zahl der Randbedingungsverletzungen durch die Reparatur jedoch stark zurück. Ein gewisser Zufallsfaktor innerhalb beider Reparaturfunktionen sorgt zudem dafür, dass nicht alle ungültigen Lösungen in die Richtung der gleichen gültigen Region gedrängt werden, sondern sich über den Lösungsraum verteilen.

5.3.4. Zusammenfassung

Zur Lösung des Problems der Optimierung von Instandsetzungsproblemen kommen verschiedene Repräsentationen in Frage. Im Rahmen dieser Arbeit wurden zwei verschiedene Repräsentationen inklusive geeigneter Operatoren für Kreuzung und Mutation entwickelt. Zudem wurden Strategien zur Behandlung ungültiger Lösungen erstellt.

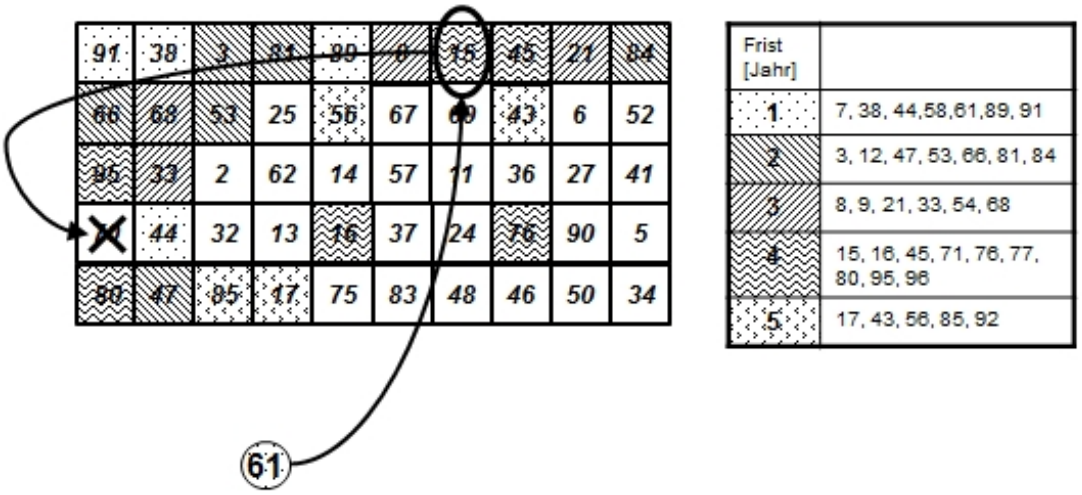

Abbildung 5.14.: Reparatur einer ungültigen Lösung in der Matrix-Repräsentation: Das kritische Bauwerk 61 mit Instandsetzungsfrist im Jahr 1 ist nicht Teil des Terminplanes. Innerhalb seiner Instandsetzungsfrist gibt es kein unkritisches Bauwerk. Das Bauwerk mit der höchsten Instandsetzungsfrist innerhalb der Frist von 61 ist das Bauwerk 15 mit einer Frist im Jahr 4. Innerhalb dieser Frist steht das unkritische Bauwerk 70. Dieses wird aus dem Terminplan entfernt, 15 nimmt dessen Platz ein, 61 den alten Platz von 15.

Eine ausführliche Untersuchung und Bewertung der beiden Repräsentationen sowie der Lösungsstrategien findet sich in Kapitel 7. Dort wird auch ein Vergleich der Eignung von Genetischen Algorithmen und Ameisenalgorithmen (Kapitel 6) für diese Problemstruktur durchgeführt.

5.4. Genetische Algorithmen für mehrkriterielle Probleme

Zur Behandlung mehrkriterieller Optimierungsprobleme, bei denen eine Menge von (pareto-)optimalen Lösungen gefunden werden soll, bieten sich Genetische Algorithmen dadurch an, dass sie anders als viele andere Optimierungsmethoden mit einer Population verschiedener Lösungen arbeiten (Deb, 2001; Coello Coello *et al.*, 2007). So können innerhalb der Population in einem einzigen Iterationsschritt mehrere Bereiche der Paretofront gleichzeitig abgesucht werden. Die Evolution der Population geschieht hier dann nicht mehr in Richtung eines einzelnen Optimums, sondern zu der Paretofront hin, auf die sich die Individuen der letzten Generation idealerweise gleichmäßig verteilen.

Die bisher gefundenen nicht-dominierten Lösungen werden dabei in der Regel in einer zweiten Population gespeichert, damit sie bei der fortschreitenden Suche

Frist [Jahr]	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	90	22	23	58	39	18	69	44	77	73	26	76	33	24	80	34	4	95	67	70	97	85	54	36	13
2	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
3	93	0	96	56	84	89	32	99	49	88	27	9	98	87	71	29	91	17	31	94	47	41	43	63	48
4	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
5	82	19	12	92	7	14	25	50	75	16	78	2	55	51	1	10	15	11	60	30	53	62	86	21	5
	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
	37	64	61	79	40	66	6	45	35	42	57	8	3	46	38	28	81	68	74	20	52	59	83	72	65

Abbildung 5.15.: Reparaturfunktion für die Permutations-Repräsentation, das Gen entspricht dem aus Abbildung 5.5. Die Schraffuren in der oberen Zeile stehen für die Fristen der Bauwerke, die in der unteren Zeile für die tatsächlichen Jahre im Terminplan. Beginnend am Anfang des Chromosoms wird an den kritischen Positionen untersucht, ob der Eintrag dort eine Instandsetzung innerhalb der jeweiligen Frist beschreibt. Ist dies nicht der Fall, wie hier für Bauwerk 3, wird die erste unkritische Position gesucht, an der ein Eintrag für eine Instandsetzung innerhalb der Frist steht, hier an der 6. Stelle. Die beiden Einträge werden dann vertauscht. An 3. Stelle steht dann der Eintrag 18, an 6. der Eintrag 23. Als nächster Schritt der Reparatur wird für Bauwerk 7 geprüft, ob sein Eintrag kleiner 10 ist (Frist von Bauwerk 7 ist das Jahr 1).

nicht wieder verloren gehen (Horn, 1996). Diese zweite Population muss dann regelmäßig mit den neu gefundenen Lösungen aktualisiert werden.

Zusätzlich zu dem Vorteil als populationsbasierter Ansatz zeigen sich Genetische Algorithmen relativ unempfindlich in Hinblick auf die Form der Paretofront und können auch Problme mit einer konkaven Paretofront behandeln. Auch diskontinuierliche Paretofronten machen ihnen keine Problme (Coello Coello *et al.*, 2007).

Einen weiteren Vorteil Genetischer Algorithmen bei der Suche nach paretooptimalen Lösungen beschreiben Louis & Rawlins (1993): Durch den Kreuzungsoperator ist es möglich, neue Individuen so zu erzeugen, dass sie die positiven Eigenschaften eines Elternteils für eine der Zielfunktionen, sowie die Eigenschaften des zweiten Elternteils, die sich positiv auf eine zweite Zielfunktion auswirken, übernehmen. Dadurch zeigen sich Genetische Algorithmen bei mehrkriteriellen Problemen anderen populationsbasierten Algorithmen (z.B. stochastische Bergsteigeralgoithmen) überlegen.

Aus genannten Gründen gibt es eine Reihe verschiedener Ansätze für mehrkriterielle Genetische Algorithmen (MOGA). Im Rahmen dieser Arbeit seien nur einige davon herausgegriffen. Für einen Überblick siehe Coello Coello (1999),Coello Coello (2000a) und Tan *et al.* (2002).

5.4.1. Non-Generational Genetic Algorithm for Multi-objective Optimization

Der Non-Generational Genetic Algorithm for Multi-objective Optimization (NGGA-MO) von Valenzuela-Rendón *et al.* (1997) ist eine Weiterentwicklung des Niched Pareto Genetic Algorithm (nPGA) von Horn *et al.* (1993, 1994).

Beiden liegt das Prinzip des *Fitness-Sharing* (Goldberg & Richardson, 1987) zugrunde. Die einzelnen (lokalen) Optima des Optimierungsproblems werden dabei als ökologische Nischen betrachtet (Conor, 1995). Individuen, die sich im Suchraum nahe beieinander befinden, konkurrieren um die Ressourcen ihrer jeweiligen Nische und werden dadurch geschwächt (vgl. Abbildung 5.16). Ein Maß für diese Schwächung ist der Nischenzähler m_i für jedes Individuum i, bei dem es sich um eine gewichtete Summe der Individuen handelt, die sich in der Nähe von i befinden:

$$m_i = \sum_{j \in \text{Pop}} s(d_{ij}) \tag{5.5}$$

mit d_{ij} dem Abstand zwischen den beiden Individuen i und j. $s(d_{ij})$ berechnet sich als

$$s(d_{ij}) = \begin{cases} 1 - (\frac{d_{ij}}{\sigma_s}) & \text{wenn } d_{ij} < \sigma_s \\ 0 & \text{sonst.} \end{cases} \tag{5.6}$$

σ_s beschreibt die Größe der Nische. In der ursprünglichen Idee des Fitness-Sharing berechnet sich dann die Fitness eines Individuums i als $f'(i) = \frac{f(i)}{m_i}$.

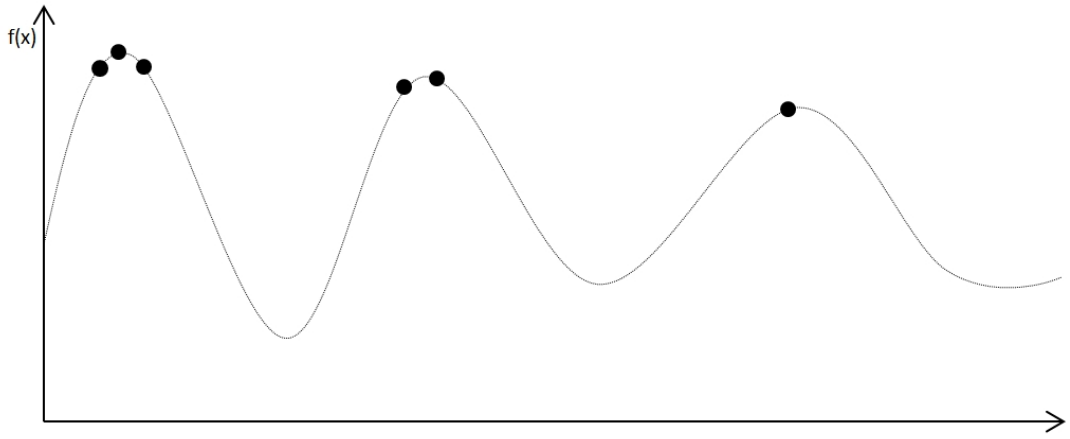

Abbildung 5.16.: Nischenbildung am eindimensionalen Beispiel: Die drei Individuen auf dem linken Optimum konkurrieren um dessen Ressourcen und werden dadurch geschwächt. Das Individuum auf dem rechten Optimum befindet sich dort allein. Nach dem Prinzip des Fitness-Sharing bekommt dieses Individuum eine höhere Wahrscheinlichkeit zur Fortpflanzung. Dadurch werden die Individuen der nächsten Generation gleichmäßiger über den Suchraum verteilt sein.

Diese Nischen eignen sich aber nicht nur dafür, die Suche von einer Konzentration auf ein einzelnes lokales Optimum abzuhalten: Bei einem mehrkriteriellen Optimierungsproblem kann der Konkurrenzkampf in den Nischen zu einer gleichmäßigeren Verteilung der Population entlang der Pareto-Front führen.

Im nPGA (Horn *et al.*, 1994) werden Individuen mit einem niedrigen Nischenzähler m bevorzugt als Eltern gewählt. Konkret wird dabei in jeder Generation die Elternmenge über Turniere zusammengestellt. Der Sieger eines Turniers zwischen zwei Individuen a und b wird folgendermaßen bestimmt:

1. Zunächst wird geprüft, ob eines der beiden Individuen das andere dominiert. Ist dies der Fall, so gewinnt das dominierende Individuum das Turnier.

2. Gibt es im ersten Schritt keinen Sieger, so wird aus der Gesamtpopulation zufällig eine Teilmenge N_1 von n_1 Individuen erstellt. a und b werden beide mit allen Individuen dieser Teilmenge verglichen. Wird eines dieser beiden von keinem der Individuen aus N_1 dominiert, das andere jedoch von mindestens einem, so ist das erste der Sieger.

3. Wenn auch in diesem zweiten Schritt kein Sieger bestimmt werden konnte, entweder da beide von keinem oder aber beide von mindestens einem der Individuen aus N_1 dominiert wurden, so wird eine weitere Teilmenge N_2 der Population aus n_2 zufälligen Elementen erstellt. Für a und b wird nun jeweils der Nischenzähler innerhalb von N_2 bestimmt. Das Individuum mit dem niedrigeren Nischenzähler ist dann Sieger des Turniers.

Der nPGA ist ein generationenbasierte Genetischer Algorithmus, bei dem die Population in jeder Generation komplett ausgetauscht wird (vgl. Abschnitt 5.2.2.4). Dies ist der Hauptunterschied zum NGGA-MO (Valenzuela-Rendón *et al.*, 1997), bei dem immer nur ein Individuum der Population, das schlechteste, durch ein neues ersetzt wird.

Zu diesem Zweck muss die implizite Fitnessfunktion des nPGA, die immer nur auf die gegenwärtige Generation bezogen ist, hier durch eine explizite Fitnessfunktion ersetzt werden, die über alle Generationen gültig bleibt. Als derartige Fitnessfunktion verwenden Valenzuela-Rendón *et al.* (1997) eine Kombination aus Dominanz, gemessen durch den Dominanzzähler δ, und einem wandernden Nischenzähler ω. Die Fitness von Individuum i berechnet sich dann als

$$f(i) = c_\delta \delta_i + c_\omega \omega_i \tag{5.7}$$

c_δ und c_ω sind dabei Wichtungsfaktoren. Das mehrkriterielle Optimierungsproblem wird dadurch auf ein zweikriterielles Minimierungsproblem reduziert.

In der Startgeneration werden zunächst die Zähler δ und ω für alle Individuen gesetzt, indem jedes Individuum i mit n_1 zufällig gewählten anderen Individuen verglichen wird. Der Dominazzähler von i δ_i entspricht dann der Anzahl an Vergleichen, in denen i dominiert wurde. Der wandernde Nischenzähler ω_i berechnet sich aus

$$\omega_i = \sum_{p=1}^{n_1} s(d_{ip}) \tag{5.8}$$

wobei sich $s(d_{ip})$ berechnet, wie in Gleichung 5.6 (vgl. Abbildung 5.17).

Nachdem alle Individuen der Startgeneration auf diese Weise einen Fitnesswert zugewiesen bekommen haben, beginnt der eigentliche Optimierungsalgorithmus. In jedem Iterationsschritt werden nun n_2-mal je zwei zufällig ausgewählte Individuen miteinander verglichen und ihre Zähler dementsprechend angepasst: Wird eines der beiden verglichenen Individuen durch das andere dominiert, so erhöht sich sein Dominanzzähler δ. Der wandernde Nischenzähler ω wird für beide Individuen entsprechend

$$\omega \leftarrow \omega - k_\omega \omega + s(d) \tag{5.9}$$

abgewandelt. Für den Parameter k_ω empfehlen Valenzuela-Rendón *et al.* (1997) den Wert $\frac{1}{n_1}$, so dass der erwartete Initialisierungswert dem Wert im Gleichgewichtszustand entspricht.

Nach der so erfolgten Aktualisierung der Zähler werden zwei Individuen entsprechend einer Fitness-Proportionalen Selektion (vgl. Abschnitt 5.2.2.1) ausgewählt und miteinander gekreuzt, so dass (gegebenenfalls nach Mutation) ein neues Individuum entsteht. Dieses neue Individuum wird in die Population aufgenommen. Dafür wird das Individuum mit dem höchsten Fitnesswert (das schlechteste Individuum) aus der Population entfernt. Anschließend werden die Zähler für das neue Individuum durch Vergleich mit n_1 zufällig gewählten Individuen der Population initialisiert(vgl. Abbildung 5.17).

```
1  Erstelle ψ zufällige Individuen als Startpopulation
2  forall the i ∈ Population do
3      δ_i ← 0
4      ω_i ← 0
5      N ← n_1 zufällig gewählte Individuen
6      forall the j ∈ N do
7          if j ≺ i then δ_i ← δ_i + 1
8          if i ≺ j then δ_j ← δ_j + 1
9          ω_i← ω_i + sh(d_ij)
10         ω_j← ω_j + sh(d_ij)
11     end
12 end
13 repeat
14     for i ← 1 to n_2 do
15         i ← zufällig gewähltes Individuum
16         j ← zufällig gewähltes Individuum
17         if j ≺ i then δ_i ← δ_i + 1
18         if i ≺ j then δ_j ← δ_j + 1
19         ω_i← ω_i − k_ω ω_i + sh(d_ij)
20         ω_j← ω_j − k_ω ω_i + sh(d_ij)
21         f_i ← c_δ δ_i + c_ω ω_i
22         f_j ← c_δ δ_j + c_ω ω_j
23     end
24     f_max ← max(f_i)
25     Wähle zwei Eltern proportional zu (f_max-f_i)
26     Erzeuge neues Individuum durch Kreuzung und Mutation
27     Lösche das Individuum mit f_i=f_max
28     neu ← neues Individuum
29     N ← n_1 zufällig gewählte Individuen
30     forall the j ∈ N do
31         if j ≺ neu then δ_neu ← δ_neu + 1
32         if neu ≺ j then δ_j ← δ_j + 1
33         ω_neu← ω_neu + sh(d_neu,j)
34         ω_j← ω_j + sh(d_neu,j)
35     end
36 until Abbruchkriterium erfüllt
```

Abbildung 5.17.: NGGA-MO

Der NGGA-MO hat den Vorteil, dass er die gesamte Paretofront gleichmäßig absucht, ohne bestimmte Regionen zu bevorzugen. Dies kann man, wenn die Gewichtung der einzelnen Optimierungskriterien für den Nutzer bekannt sind, auch als Nachteil betrachten. Auch eine Interaktion mit dem Nutzer während der Optimierung ist so nicht möglich. Sind die Wünsche des Nutzers allerdings nicht bekannt und soll diesem für die Auswahl der bevorzugten Lösung ein möglichst breites Spektrum verschiedener Lösungen angeboten werden, so ist diese Eigenschaft durchaus als Vorteil zu betrachten.

Als weiterer Vorteil ist zu nennen, dass NGGA-MO mit einer recht geringen Anzahl an Funktionsauswertungen auskommt: Für die in Valenzuela-Rendón *et al.* (1997) behandelten drei Probleme fanden im Vergleich zum nPGA bei gleicher Abdeckung der Pareto-Front nur bis zu 1/10 an Funktionsauswertungen statt.

Der große Nachteil des NGGA-MO ist die große Menge an benötigten Parametern: Sowohl die Anzahl an Vergleichoperationen für beide Fälle, n_1 und n_2, als auch die beiden Wichtungsparameter c_δ und c_ω müssen gesetzt werden. Für keinen dieser Parameter liefern Valenzuela-Rendón *et al.* (1997) einen Richtwert und zumindest die Wichtungsparameter scheinen stark problemabhängig zu sein.

5.4.2. Ansatz von Jaszkiewicz

Einen grundsätzlich anderen Ansatz verfolgt Jaszkiewicz (2002). Selber bezeichnet er seinen Algorithmus als MOGLS (*multi-objective genetic local search*, multikriterielle genetische lokale Suche[19]).

Der in Jaszkiewicz (2002) beschriebene Algorithmus ist eine Erweiterung des Algorithmus von Ishibuchi & Murata (1998). Beide basieren darauf, dass die Pareto-Front eines n-kriteriellen Optimierungsproblems als die Menge aller Punkte beschrieben werden kann, die durch die Optimierung der Funktion

$$f(x) = w_1 * f_1(x) + \ldots + w_i * f_i(x) + \ldots + w_n * f_n(x) \tag{5.10}$$

mit allen möglichen $w_1, \ldots, w_n \in [0; 1]$ erreicht werden (Murata & Ishibuchi, 1995). Anders ausgedrückt ist die Pareto-Front die Menge der Optima aller möglichen gewichteten Summen der einzelnen Optimierungskriterien (siehe auch Abschnitt 2.5.2). Eine wiederholte Optimierung mit immer auf's Neue zufällig gewählten Faktoren $w_1, \ldots, w_n$ deckt also die gesamte Pareto-Front ab.

Bei dem Algorithmus von Ishibuchi & Murata (1998) handelt es sich um einen generationenbasierten Ansatz, bei dem die gesamte Population in jeder Generation komplett ausgetauscht wird. Es werden zwei Mengen von Lösungen vorgehalten: die aktuelle Population sowie eine Menge an bisher nicht-dominierten Lösungen. Die Population wird mit zufällig erstellten Lösungen initialisiert. Aus diesen

[19] Genetische lokale Suche ist ein Synonym zu memetischen Algorithmen.

werden die nicht-dominierten Lösungen ausgewählt und in die Menge der nicht-dominierten Lösungen N_{nonDom} kopiert. Danach finden in jeder Generation die folgenden Schritte statt (vgl. Abbildung 5.18):

1. $\psi - n_{\text{elite}}$ Elternpaare werden über eine fitnessproportionale Selektion ausgewählt. Die Fitness der einzelnen Individuen wird nach Gleichung 5.10 bestimmt, wobei die Wichtungsfaktoren $w_1, \ldots, w_n$ für jedes Elternpaar individuell zufällig gesetzt werden.

2. Für jedes der Elternpaare aus Schritt 1 wird durch Kreuzung ein Kind erzeugt. Dieses wird zusätzlich einer Mutation unterworfen. Die so erzeugten neuen Individuen werden dann in die Population für die nächste Generation übernommen.

3. Aus N_{nonDom} werden n_{elite} Lösungen zufällig ausgewählt und ebenfalls in die neue Population kopiert.

4. Alle Individuen der neuen Population werden einer lokalen Suche unterworfen. Als Wichtungsfaktoren werden dabei für jedes Individuum diejenigen verwendet, die zur Auswahl seiner Eltern benutzt wurden[20].

5. N_{nonDom} wird mit den Lösungen der neuen Population aktualisiert. Das heißt, neue Lösungen, die von keiner der Lösungen aus N_{nonDom} dominiert werden, werden neu aufgenommen, Lösungen aus N_{nonDom}, die von diesen neuen Lösungen dominiert werden, aus der Menge entfernt.

Jaszkiewicz (2002) übernimmt große Teile dieses Konzepts, allerdings verwendet er einen nicht-generationenbasierten Ansatz, bei dem in jeder Iteration nur ein neues Individuum erstellt wird. Damit erspart er sich das Einführen einer Elite-Regel. Als weitere Änderung verwendet er ein strengeres Selektionskriterium zur Auswahl der Elternpaare, was die Suche stärker in Richtung der jeweilig aktuellen gewichten Fitnessfunktion beeinflusst: Nur die K (für zufällig erstellte Wichtungsfaktoren) besten Individuen werden in eine separate Elternmenge kopiert. Aus dieser Elternmenge werden zwei Individuen zufällig gezogen und miteinander gekreuzt, um ein neues Individuum zu erzeugen. Dieses wird dann einer Mutation und einer lokalen Suche unterworfen. K bestimmt dabei die Stärke des Selektionsdrucks.

Die Größe der Population ist bei Jaszkiewicz variabel: Für die Startpopulation werden S Individuen zufällig erzeugt. Wird bei der Kreuzung ein Individuum erzeugt, das bezüglich der gewichteten Fitnessfunktion, nach der die Elternmenge ausgewählt wurde, besser ist als alle Individuen der Elternmenge, so wird es zur Population hinzu gefügt. Erreicht die Population eine Größe von $K * S$, so wird das älteste Individuum aus ihr entfernt, die Population wird also behandelt wie eine FIFO-Schlange.

[20] Ishibuchi & Murata (1998) geben keine Auskunft darüber, welche Wichtungsfaktoren für die lokale Suche bei den aus der Menge der nicht-dominierten Lösungen kopierten Elitelösungen verwendet wird.

```
1  repeat
2  |  forall the i ∈ Pop do
3  |  |  bestimme zufällig w_1 ... w_n
4  |  |  f(i)←w_1 * f_1(i) + ...+w_n * f_n(i)
5  |  end
6  |  for i← 1 to ψ − n_elite do
7  |  |  wähle zwei Individuen x_1 und x_2 aus Pop proportional zu f(x)
8  |  |  x_3 ← kreuze x_1 und x_2
9  |  |  Mutation an x_3
10 |  |  Kopiere x_3 in neue Population Pop_neu
11 |  end
12 |  for i← 1 to n_elite do
13 |  |  Kopiere zufälliges Individuum aus N_nonDom in Pop_neu
14 |  end
15 |  Population ←Pop_neu
16 |  forall the i ∈ Pop do
17 |  |  Lokale Suche an Individuum i
18 |  end
19 |  Aktualisiere N_nonDom
20 until Abbruchkriterium erfüllt
```

Abbildung 5.18.: Algorithmus von (Ishibuchi & Murata, 1998) - Hauptschleife

```
1  repeat
2  |  bestimme zufällig w_1 ... w_n
3  |  forall the i ∈ CS do
4  |  |  f(i)←w_1·f_1(i) + ...+w_n·f_n(i)
5  |  end
6  |  Wähle die K besten Individuen aus CS und kopiere sie in TP
7  |  Wähle zufällig zwei Individuen x_1 und x_2 aus TP
8  |  x_3 ← kreuze x_1 und x_2
9  |  x'_3 ←Lokale Suche an x_3
10 |  if f(x'_3) > f(i) ∀ i ∈ TP then
11 |  |  Füge x'_3 zu CS hinzu
12 |  |  Aktualisiere N_nonDom
13 |  end
14 until Abbruchkriterium erfüllt
```

Abbildung 5.19.: Algorithmus von (Jaszkiewicz, 2002) - Hauptschleife

Zusammengefasst umfasst eine Iteration in dem MOGLS nach Jaszkiewicz die folgenden Schritte (vgl. Abbildung 5.19):

1. Eine Fitnessfunktion entsprechend Gleichung 5.10 wird durch Setzen zufälliger Wichtungsfaktoren erstellt.

2. Aus der Gesamtpopulation CS werden die K entsprechend dieser Fitnessfunktion besten Individuen ausgewählt und in eine Elternmenge TP kopiert.

3. Aus TP werden zwei zufällige Individuen x_1 und x_2 gewählt und miteinander gekreuzt, so dass ein neues Individuum x_3 entsteht.

4. An x_3 wird eine lokale Suche entsprechend der derzeit gültigen Fitnessfunktion durchgeführt.

5. Das daraus resultierende Individuum x'_3 wird mit allen Individuen aus TP verglichen. Ist es entsprechend der aktuellen Fitnessfunktion besser als alle diese, so wird es zu CS hinzugefügt. Die Menge der nicht-dominierten Lösungen wird entsprechend aktualisiert.

Der MOGLS hat als nicht-generationenbasierter Genetischer Algorithmus den Vorteil, dass er mit verhältnismäßig wenig Funktionsauswertungen auskommt, um gute Ergebnisse zu erreichen. Jaszkiewicz (2002) testete den Algorithmus im Vergleich mit anderen multikriteriellen Genetischen Algorithmen (unter anderem dem von Ishibuchi & Murata (1998)) auf drei verschiedenen multikriteriellen TSP-Instanzen und konnte dort zeigen, dass die Qualität der gefundenen Lösungen bei seinem MOGLS höher war als bei den verglichenen Algorithmen, gleichzeitig aber nur einen Bruchteil der Rechenzeit benötigte.

Als nachteilig am MOGLS ist anzumerken, dass auch dieser mehrere zusätzliche Parameter benötigt, vor allem die Größe der Startpopulation S sowie die der Elternmenge K. Auch die zulässige Größe der Population wurde von Jaszkiewicz (2002) nur rein empirisch mit $K * S$ als die Größe bestimmt, bei der keine Qualitätsunterschiede gegenüber einem Durchlauf, bei dem alle Lösungen behalten wurden, mehr auftraten.

Auch für die Bestimmung von S führt Jaszkiewicz (2002) eine Heuristik ein. Danach wird die Startpopulation so lange vergrößert, bis die durchschnittliche Qualität der K besten Lösungen für eine beliebige Fitnessfunktion nach Gleichung 5.10 der durchschnittlichen Qualität der gesamten Startpopulation entspricht. Dieser Ansatz zeigt sich jedoch bei einer hohen Rechenzeit für die Auswertung der Zielfunktionen, wie beim Einfluss auf den Verkehr im Instandsetzungsproblem, als unpraktikabel, da es zu sehr großen Startpopulationen führen kann (bis zu 1300 Individuen beim drei-kriteriellen TSP in Jaszkiewicz (2002)). Damit verbraucht allein die Generierung der Startpopulation so viel Rechenzeit wie sonst ein gesamter Durchlauf der Optimierung. Daher wird in dieser Arbeit eine kleinere, statische Populationsgröße verwendet.

Da die lokale Suche für das Instandsetzungsproblem durch die sehr große Nachbarschaft einer Lösung sehr rechenintensiv ist und die Gefahr besteht, dass diese der Hauptsuchmechanismus des Algorithmus wird, wird in dieser Arbeit eine modifizierte Version des Algorithmus von Jaszkiewicz verwendet. In diesem, zukünftig in Anlehnung an den Namen Jaskiewicz als MOGA-Jas bezeichneten Algorithmus wird das Fortpflanzungskonzept des MOGLS übernommen. Statt einer lokalen Suche wird ein durch Kreuzung neu erzeugtes Individuum allerdings nur einer (nicht zielgerichteten) Mutation, wie in den Abschnitten 5.3.2.1 bzw. 5.3.2.2 beschrieben, unterworfen.

5.4.3. Zusätzliche Ergänzungen für das mehrkriterielle Instandsetzungsproblem

Wie auch das einkriterielle Instandsetzungsproblem, so ist auch das mehrkriterielle stark beschränkt. Da die selben Repräsentationen verwendet werden, ist zudem der gültige Lösungsraum auch hier nicht zusammenhängend.

Weder die Formulierung des nPGA noch die des MOGA-Jas sieht ursprünglich Randbedingungen vor. Methoden zu deren Behandlung müssen also zusätzlich eingeführt werden.

Grundsätzlich können die in Abschnitt 5.3.3 für das einkriterielle Problem beschriebenen Methoden auch auf das mehrkriterielle Problem angewandt werden. Dabei sind jedoch folgende Punkte zu beachten:

- Anders als beim einkriteriellen Problem genügt es nicht, eine einzige gültige Lösung in der Startpopulation zu besitzen. Stattdessen sollte die Startpo-

pulation ein möglichst breit gefächertes Spektrum gültiger Lösungen enthalten, um bei der Suche möglichst die ganze Front abzudecken. Aus diesem Grund wird die vorgeschaltete Suche mit der Minimierung der Randbedingungsverletzungen als einzigem Zielkriterium nicht, wie beim einkritriellen Problem, abgebrochen, sobald die erste gültige Lösung erreicht wurde, sondern erst, wenn ψ bzw. S gültige Lösungen gefunden wurden. Diese bilden dann die Startpopulation.

- Nur gültige Lösungen werden in die Menge der nicht-dominierten Lösungen übernommen.
- Strafen lassen sich nur beim MOGA-Jas sinnvoll einsetzen, wo sie auf die gewichtete Fitnessfunktion aufaddiert werden. Für den NGGA-MO bietet sich dagegen ein anderes Vorgehen an. Hier kann die Gültigkeit bzw. Ungültigkeit einer Lösung in den Dominanzzähler δ eingehen: Eine ungültige Lösung wird von einer gültigen Lösung immer dominiert, egal welche einzelnen Zielfunktionswerte die beiden Lösungen haben. Nur wenn beide Lösungen gültig oder beide Lösungen ungültig sind, wird die Dominanz entsprechend dem üblichen Dominanzkriterium bestimmt.
- Reparaturfunktionen können im mehrkriteriellen Fall genau so wie im einkriteriellen Fall verwendet werden.

5.4.4. Zusammenfassung

Für die Behandlung des mehrkriteriellen Instandsetzungsproblems kommen verschiedene mehrkriterielle Genetische Algorithmen in Frage. In dieser Arbeit untersucht wurden der Non-Generational Genetic Algorithm for Multi-objective Optimization (NGGA-MO) von Valenzuela-Rendón *et al.* (1997) sowie eine abgewandelte Version des Algorithmus von Jaszkiewicz (2002).

Für beide Ansätze sind zusätzliche Anpassungen notwendig, um ungültige Lösungen sinnvoll behandeln zu können. Diese orientieren sich im wesentlichen an den für den einkriteriellen Fall vorgeschlagenen Maßnahmen.

6. Ameisenalgorithmen

6.1. Grundlagen

Ameisenalgorithmen (englisch: Ant Colony Optimization, ACO) sind eine relativ neue Metaheuristik. Entwickelt wurden sie von Dorigo zu Beginn der 1990er Jahre (Dorigo, 1992). Die grundsätzliche Idee basiert auf dem Verhalten einiger Ameisenarten, z.B. der Argentinischen Ameise *Iridomyrmex humilis* (Goss *et al.*, 1989; Deneubourg *et al.*, 1990) und der Schwarzen Wegameise *Lasius niger* (Bonabeau *et al.*, 1997). Diese Ameisen sind zur Selbstorganisation fähig, um den kürzesten Weg zwischen ihrem Nest und einer Futterquelle zu finden.

In den Zweibrücken-Experimenten, wie sie von Goss *et al.* (1989) beschrieben werden, ließ sich die Ursache dafür ergründen.

In einem ersten Experiment wurden verschiedenen Kolonien von *Iridomyrmex humilis* jeweils zwei gleich lange Brücken als Weg zwischen ihrem Bau und einer

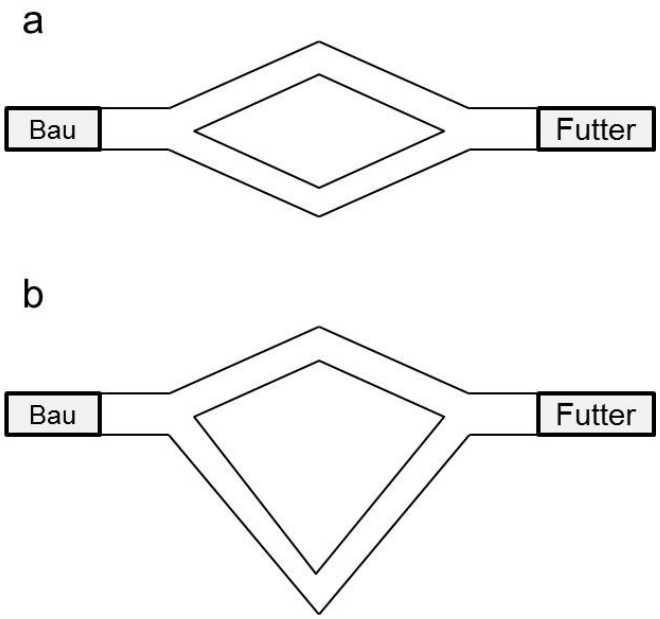

Abbildung 6.1.: Versuchsaufbau des Zweibrücken-Experiments: **a** mit zwei gleich langen Brücken, **b** mit Brücken unterschiedlicher Länge. (Abbildung nach Dorigo & Stützle (2004))

Futterquelle zur Verfügung gestellt (vgl. Abbildung 6.1a). Dann wurde über die Zeit gestaffelt beobachtet, von wie vielen Ameisen welche Brücke gewählt wurde. Nach anfänglich zufälliger Wahl des Weges bildete sich nach einiger Zeit in allen Experimenten ein bevorzugter Pfad aus. Dieser lag über alle Versuche gesehen etwa gleich häufig auf beiden Brücken.

Dieses Verhalten lässt sich damit erklären, wie die Ameisen sich orientieren: Beim Laufen sondern die Tiere einen Duftstoff, Pheromon, ab. Nachfolgende Ameisen wählen bevorzugt Wege, auf denen bereits Pheromon liegt. Je mehr Pheromon auf einem Weg liegt, desto wahrscheinlicher ist die Wahl dieses Weges. Da die neue Ameise wiederum Pheromon ablagert, wirkt dieser Effekt selbstverstärkend. Für das Zweibrücken-Experiment bedeutet dies, dass eine zufällige Schwankung der Pheromonmenge auf den beiden Brücken dazu führt, dass die Brücke mit einer geringfügig höheren Menge bevorzugt gewählt wird. Dabei wird zusätzliches Pheromon auf dieser Brücke abgelagert, so dass das Verhältnis der Pheromonmengen auf beiden Brücken sich weiter verschiebt. Dies wiederholt sich, bis irgendwann auf einer Brücke so viel Pheromon ist, dass nur noch diese gewählt wird.

Wie sich dieses Phänomen bei der Übertragung auf Optimierungsprobleme auswirken kann, zeigt ein zweites Experiment. Wieder wurde der Bau der Ameisen über zwei Brücken mit einer Futterquelle verbunden. Allerdings waren diese Brücken nicht mehr gleich lang, sondern eine der beiden Brücken hatte die doppelte Länge der anderen (vgl. Abbildung 6.1b). Hier zeigte sich, dass in fast allen Versuchen nach einiger Zeit alle Ameisen den kürzeren Weg wählten.

Die Erklärung liegt wiederum in der Pheromonablagerung: Anfangs ist auf keiner der beiden Routen Pheromon vorhanden, die Routenwahl erfolgt also zufällig. Wählt nun eine Ameise den kürzeren Weg, erreicht sie eher das Futter als eine Ameise auf dem längeren Weg. Sie sammelt Futter und steht nun vor der Entscheidung, welche Route sie für den Rückweg wählt. Da auf dem Weg, auf dem sie gekommen ist, schon Pheromon liegt, auf der Alternative aber keines, kehrt sie auf demselben Weg zurück und lagert dabei dort zusätzliches Pheromon ab. Eine weitere Ameise, die nun, egal auf welcher Route, an der Futterquelle ankommt, findet nun auf der kürzeren Route mehr Pheromon vor und wird diese nun bevorzugt wählen, wodurch sich die Attraktivität dieser Route weiter verstärkt.

Ein drittes Experiment (Dorigo & Stützle, 2004) zeigt einen weiteren wichtigen Bestandteil des Systems. Diesmal wird den Ameisen zu Beginn nur eine Brücke zur Futterquelle zur Verfügung gestellt. Nach 30 Minuten, also nachdem sich bereits eine Pheromonspur ausgebildet hat, wird eine zweite, kürzere, Brücke dazugefügt. Ein Großteil der Ameisen wählt in den meisten Versuchen dann weiterhin die längere Route. Allerdings finden sich einige Ameisen, die die kürzere Route wählen, in einigen Versuchen sogar die Mehrheit.

Erklärbar ist das Verharren auf der längeren Route durch die hohe Konzentration an Pheromon auf dieser. Dieses verdunstet zu langsam, als dass die neue Route attraktiv würde. Dadurch wird die Konzentration auf der alten Route noch

weiter erhöht. Wäre die Verdunstungsrate allerdings höher, so könnte die Kolonie die einmal „gelernte“ schlechtere Route „vergessen“ und sich so an die neuen Gegebenheiten anpassen[1] (Dorigo & Stützle, 2004).

6.2. Algorithmus

6.2.1. Ant System

Die erste Implementierung von Ameisenalgorithmen zur Lösung von Optimierungsproblemen war das *Ant System (AS)* von Dorigo (Dorigo *et al.*, 1991; Dorigo, 1992; Dorigo *et al.*, 1996). Sie erfolgte für das Problem des Handlungsreisenden (TSP, vgl. Abschnitt 2.2). Zum besseren Verständnis wird die Methode, ebenso wie ihre Abwandlungen in den folgenden Abschnitten, hier ebenfalls für das TSP beschrieben, mit entsprechenden Änderungen ist es aber auch auf andere Optimierungsprobleme übertragbar. Die grundlegenden Abläufe sind auch in allen späteren Versionen von Ameisenalgorithmen die selben (Dorigo & Stützle, 2004; Boysen, 2010).

In Anlehnung an das Verhalten echter Ameisen wie in Abschnitt 6.1 beschrieben, konstruieren künstliche Ameisen einen Weg, der eine mögliche Lösung des Problems beschreibt (beim TSP ist dies eine Route durch alle Städte). Anders als beim natürlichen Vorbild, wo sich die Ameisen in einem kontinuierlichen Raum bewegen, ist der Suchraum für die künstlichen Ameisen notgedrungen diskret. Das Problem wird daher als Graph formuliert; eine Route durch diesen Konstruktionsgraph entspricht einer möglichen Lösung des Problems. Für das TSP, das von sich aus schon als Graph beschrieben ist, kann dieser Problemgraph gleich als Konstruktionsgraph verwendet werden (Dorigo & Stützle, 2004).

Bei der Konstruktion eines Weges orientieren die künstlichen Ameisen sich an dem Pheromon, welches auf den Kanten abgelagert liegt. Zusätzlich bekommen die Ameisen noch eine heuristische Information[2]. Eine Ameise k, die in einer Stadt i steht, entscheidet sich demnach mit der Wahrscheinlichkeit

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha \cdot [\eta_{il}]^\beta} \qquad \text{, für } j \in N_i^k \tag{6.1}$$

dafür, als nächstes Stadt j zu besuchen.

[1] bzw. auf eine von Anfang an existierende bessere Route ausweichen, die anfangs weniger attraktiv erschien, jedoch später zufällig gefunden wurde

[2] Man kann dies so interpretieren, dass sich die Ameise nicht nur an ihrem Geruchssinn orientiert, sondern zusätzlich auch über eine gewisse Sehfähigkeit verfügt (Boysen, 2010).

Dabei ist τ_{ij} die Pheromonmenge auf der Kante zwischen i und j. η_{ij} ist die entsprechende heuristische Information[3]. Die Menge N_i^k enthält alle Städte, die Ameise k noch nicht besucht hat. α und β sind frei wählbare Parameter zur relativen Gewichtung zwischen Pheromon und heuristischer Information.

Ein $\alpha = 0$ entspricht dabei einem Greedy-Ansatz, bei dem immer die nächstgelegene Stadt mit der höchsten Wahrscheinlichkeit gewählt wird. Dagegen bedeutet ein $\beta = 0$, dass die Ameisen sich rein an den vorhandenen Pheromonspuren orientieren. In Dorigo *et al.* (1996) wird gezeigt, dass dies zu einer verfrühten Stagnation führt, das heißt, alle Ameisen wählen den selben suboptimalen Pfad. Ein zu großes β wirkt sich dagegen negativ auf die Konvergenzgeschwindigkeit aus (Ning *et al.*, 2010). Welche Werte für α und β ideal sind, ist problemabhängig. Für das TSP empfehlen Dorigo & Stützle (2004) $\alpha = 1$ und $2 \leq \beta \leq 5$. Merkle *et al.* (2002) schlagen vor, den Wert für β mit jeder Generation zu reduzieren, so dass die heuristische Information mit der Zeit an Gewicht verliert.

Bei anderen Problemen als dem TSP kann es sich als sinnvoll erweisen, neben η zusätzliche heuristische Parameter als Faktoren in Gleichung 6.1 einzuführen, die der Optimierungsgröße nahestehene Eigenschaften des Problems beschreiben (vgl. hierzu z.B. Bullnheimer *et al.* (1997b) und Gagné *et al.* (2001)).

Außerdem können den Ameisen noch zusätzliche Informationen mitgegeben werden, indem für jede mögliche Entscheidung eine Abschätzung entsprechend dem Branch-and-Bound-Verfahren vorgenommen wird, wie sich diese auf die Lösungsqualität auswirkt und diese als Faktor in Gleichung 6.1 eingehen lässt (Gagné *et al.*, 2001). Dies kann zu schnellerer Konvergenz und besseren Lösungen führen, ist aber natürlich nur dann sinnvoll, wenn man eine derartige Abschätzung unter reellem Aufwand treffen kann, ohne in einer Vollständigen Enumeration zu enden. Maniezzo (1999) und Maniezzo & Colorni (1999) verwenden in ihrer Anwendung des *Ant System* auf das quadratische Zuordnungsproblem[4] allein die Abschätzung als heuristische Information η.

Zusätzliches Wissen kann den Ameisen auch über eine *Kandidatenliste* mitgegeben werden (vgl. Dorigo & Gambardella (1997)). In dieser werden die jeweils besten Wahlmöglichkeiten, basierend auf einer Heuristik (beim TSP z.B. die Entfernung zwischen zwei Städten) in jedem Punkt vorgehalten und nur diese werden in die Menge N_i^k in Gleichung 6.1 aufgenommen. Diese Liste kann am Anfang einmal erstellt und dann für den gesamten Optimierungsprozess weiterverwendet werden. Nur wenn alle Möglichkeiten aus der *Kandidatenliste* tabu sind, erfolgt eine Wahl aus allen verfügbaren Möglichkeiten. Ziel dabei ist es, den Suchpro-

[3]Für das TSP wählt man meistens $\eta_{ij} = 1/d_{ij}$, wobei d_{ij} die Entfernung zwischen beiden Städten beschreibt.

[4]Bei dem quadratischen Zuordnungsproblem sollen n Aktivitäten (z.B. Vorlesungen) so auf n Räumlichkeiten (z.B. Hörsäle) verteilt werden, dass das Produkt aus Fluss (z.B. Anzahl der Studenten, die von einer Vorlesung zur nächsten gehen) und zurückgelegter Strecke minimal ist. Dieses Problem tritt beispielsweise auch beim Entwurf von elektronischen Schaltkreisen und Mikrochips auf.

zess zu beschleunigen. Gambardella & Dorigo (1996) zeigen für das TSP eine Verbesserung der Ergebnisse durch Verwendung einer *Kandidatenliste*. Ähnliches zeigen Bullnheimer *et al.* (1999) für das Problem der Tourenplanung (engl. *Vehicle Routing Problem*, VRP), wobei hier nicht klar ist, wie viel der Verbesserung auf die anderen veränderten Eigenschaften des Algorithmus (andere Definition von η, Verwendung von Rank-Based Ant System (siehe Abschnitt 6.2.3)) zurückzuführen ist.

Anders als bei ihrem natürlichen Vorbild erfolgt die Pheromonablagerung der künstlichen Ameisen nicht während der Konstruktion der Routen. Sämtliche Aktionen, die das Pheromon betreffen, werden im ursprünglichen *Ant System* am Ende eines Iterationsschrittes gemeinsam ausgeführt.

Nachdem alle Ameisen der Kolonie eine vollständige Lösung konstruiert haben, wird zunächst das Pheromon auf allen Kanten entsprechend

$$\tau_{ij} \leftarrow (1-\rho) * \tau_{ij} \qquad ,\forall (i,j) \in L \tag{6.2}$$

„verdunstet". $0 < \tau \leq 1$ ist dabei der *Verdunstungsgrad*. Auch dieser Parameter muss dem Problem angepasst werden. Für das TSP empfehlen Dorigo & Stützle (2004) $\tau = 0.5$.

Die Verdunstung ist nötig, um schlechte Lösungen zu „vergessen" (vgl. Abschnitt 6.1, Cordón García *et al.* (2002)). Außerdem verhindert sie eine unbegrenzte Ablagerung von Pheromon auf einzelnen Kanten (Dorigo & Stützle, 2004) und stärkt den Einfluss neuerer Lösungen, die der Heuristik entsprechend besser sind als ältere (Boysen, 2010).

Nach der Verdunstung lagern alle Ameisen k der Kolonie entsprechend der Länge C^k des von ihnen gefundenen Weges T^k auf allen von ihnen verwendeten Kanten Pheromon ab. Die Menge $\Delta\tau_{ij}^k$ berechnet sich dabei aus

$$\Delta\tau_{ij}^k = \begin{cases} 1/C^k & \text{wenn die Kante}(i,j) \in T^k \\ 0 & \text{sonst.} \end{cases} \tag{6.3}$$

Solimanpur *et al.* (2005) beschreiben eine alternative Art der Berechnung von $\Delta\tau_{ij}^k$. Dieses bestimmt sich danach als

$$\Delta\tau_{ij}^k = \begin{cases} \lambda * C^*/C^k & \text{wenn die Kante}(i,j) \in T^k \\ 0 & \text{sonst.} \end{cases} \tag{6.4}$$

wobei C^* die Länge der besten bisher gefundene Lösung darstellt. Bei λ handelt es sich um einen Skalierungsfaktor. Ein Vorteil dieses Ansatzes ist, dass die Größenordnung von τ immer gleich ist und nicht für jede Probleminstanz von außen neu angepasst werden muss.

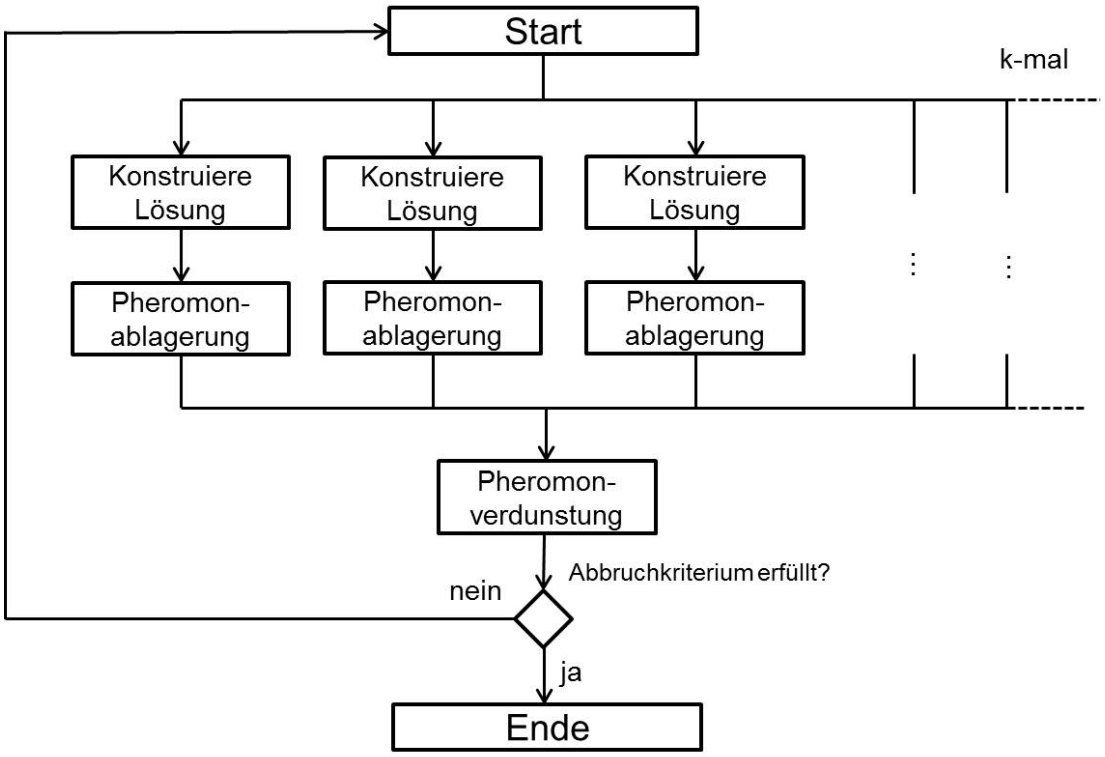

Abbildung 6.2.: Ablaufdiagramm für das Ant System.

Zusammengefasst aus den Gleichungen 6.2 und 6.3 ergibt sich für jede Kante $(i, j) \in L$:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}. \tag{6.5}$$

Nachdem die Pheromonmenge auf allen Kanten entsprechend Gleichung 6.5 abgeändert wurde, startet die nächste Generation Ameisen. Die neuen Ameisen wählen nun ihre Route wieder entsprechend Gleichung 6.1 mit den neuen Werten für τ_{ij}. Damit die Lösungen der ersten Generation, die wegen der gleichen Pheromonmenge auf allen Kanten noch starke Ähnlichkeit mit einer Greedy-Lösung aufweist, nicht zu sehr ins Gewicht fallen, müssen die Kanten mit einer Startmenge Pheromon τ_0 vorbelegt werden. Diese sollte der Größenordnung der in einem Iterationsschritt abgelagerten Pheromonmenge entsprechen. Zu deren Berechnung kann die Länge einer mit einem Greedy-Algorithmus oder einer anderen Heuristik berechneten Route herangezogen werden (Dorigo & Stützle, 2004). In Abbildung 6.2 ist der generelle Ablauf der Optimierung für das Ant System noch einmal dargestellt.

6.2.2. Elitist Ant System

Das *Ant System*, wie in Abschnitt 6.2.1 beschrieben, hat verschiedene Nachteile. Unter anderem kann der Algorithmus einmal gefundene gute Lösungen wieder vergessen, wenn diese nur von einzelnen Ameisen gefunden wurden. Dies führt nicht nur zum Verlust dieser einen Lösung[5] sondern ganzer eventuell vielversprechender Bereiche des Suchraums in ihrer Nachbarschaft.

Ein erster Ansatz, dieses Problem zu überwinden, wurde in Dorigo (1992) und Dorigo *et al.* (1996) beschrieben. Das *Elitist Ant System* bildet eine Erweiterung des ursprünglichen *Ant System* um eine Dämonenaktivität[6].

Die Routenwahl der einzelnen Ameisen erfolgt analog zum *Ant System*. Allerdings wird eine zusätzliche Ameise, die *Elitist Ant*, vorgehalten. Diese konstruiert selber keine Route, sondern speichert die beste bisher gefundene Lösung. Diese bleibt so lange unverändert, bis eine noch bessere Lösung gefunden wurde (vgl. Abbildung 6.3). Bei der Pheromonablagerung wird nun auch von der *Elitist Ant* Pheromon abgelagert, die Menge errechnet sich entsprechend Gleichung 6.3. Um dieser Elitist-Lösung zusätzlich Gewicht zu verleihen, wird sie um einen Faktor e verstärkt. Insgesamt ergibt sich dann für das Pheromon-Update

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + e\Delta\tau_{ij}^{\text{elitist}} \tag{6.6}$$

.

In Dorigo (1992) und Dorigo *et al.* (1996) wurde gezeigt, dass mit diesem Ansatz, bei geeigneter Wahl des Parameters e[7], für verschiedene Ausprägungen des TSP bessere Lösungen gefunden wurden als mit dem ursprünglichen *Ant System*. Außerdem benötigt das *Elitist Ant System* weniger Iterationen. Ein Vergleich mit Genetischen Algorithmen und Simulated Annealing in Bullnheimer *et al.* (1997a) über verschieden große Instanzen des TSP zeigt eine vergleichbare Güte der Lösungen bei kleinen Probleminstanzen. Bei größeren Instanzen zeigte sich das *Elitist Ant System* den beiden anderen Methoden leicht überlegen.

Um eine vorzeitige Konvergenz auf die Elitist-Lösung zu verhindern, schlagen Merkle *et al.* (2002) vor, nur eine gewisse Anzahl g_{max} von Generationen die gleiche Elitist-Lösung zuzulassen. In der nächsten Iteration wird die Elitist Ant durch die beste Lösung dieser Iteration ersetzt, selbst wenn diese schlechter ist als die bisherige Elitist Ant. Dies ergibt vor allem dann Vorteile, wenn in der Nachbarschaft der bisherigen Elitist-Lösung keine weiteren guten Lösungen liegen.

[5] Diese könnte man auch so vorhalten, bis eine bessere gefunden wurde, so dass das Endergebnis immer die über sämtliche Iterationsschritte beste gefundene Lösung ist.

[6] Mit Dämonenaktivitäten werden Aktivitäten im Algorithmus bezeichnet, die nicht von einer einzelnen Ameise ausgeführt werden können, sondern zentral gesteuert werden müssen (Dorigo & Stützle, 2004).

[7] Für das TSP empfehlen Dorigo & Stützle (2004) $e = n$.

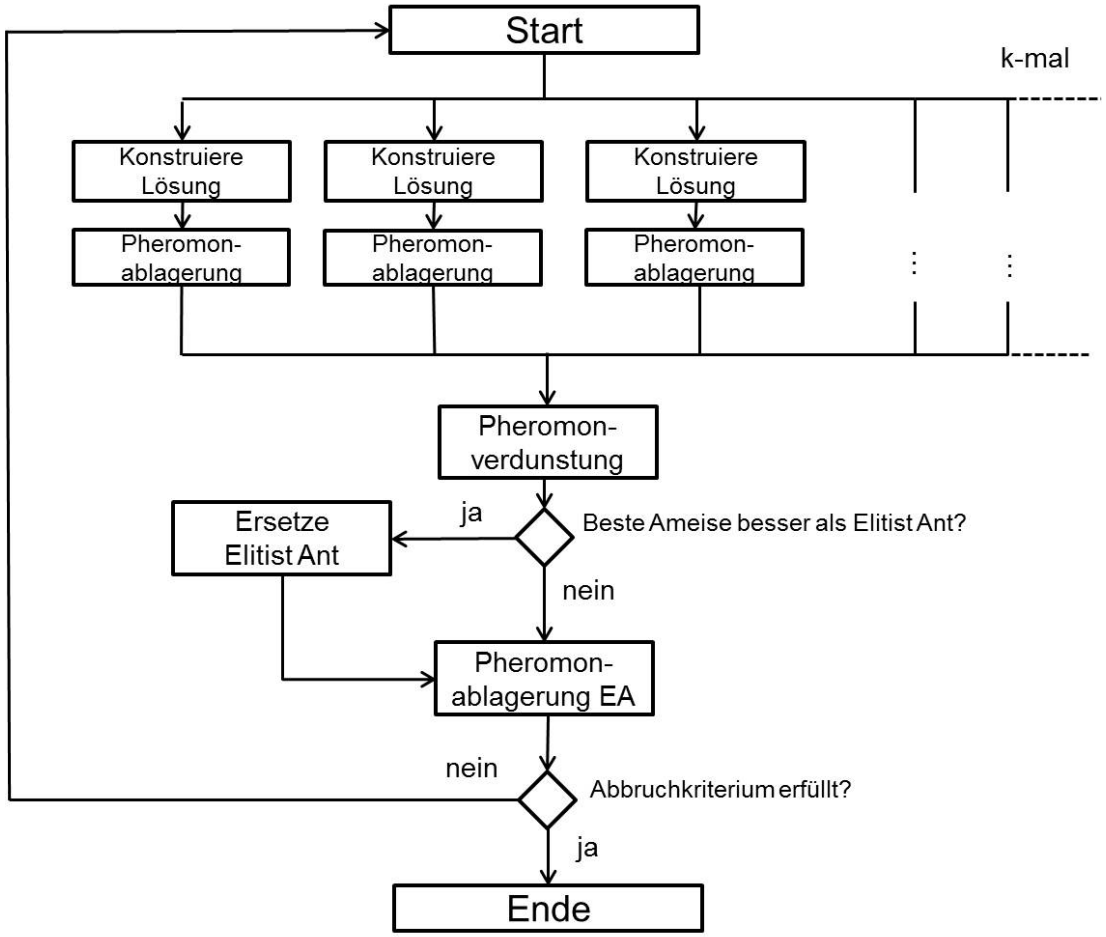

Abbildung 6.3.: Ablauf des Elitist Ant System

6.2.3. Rank-Based Ant System

Eine weitere Variation des *Ant System*, die versucht dessen Nachteile zu überwinden, das *Rank-Based Ant System*, stammt von Bullnheimer *et al.* (1997a). Wie im *Elitist Ant System* wird auch hier eine zusätzliche Ameise vorgehalten, in der die beste bisher gefundene Lösung gespeichert wird.

Anders als im ursprünglichen *Ant System* und im *Elitist Ant System* ist die abgelagerte Pheromonmenge hier aber nicht mehr nur von der absoluten Qualität der gefundenen Lösung[8] abhängig, zusätzlich geht auch noch ein, wie sich die Qualität der Lösung zu den anderen in diesem Iterationsschritt gefundenen Lösungen verhält, d.h. welchen Rang[9] die betreffende Ameise in der Kolonie hat.

Nur eine begrenzte Anzahl an Ameisen, die $(w-1)$ besten, lagert überhaupt Pheromon ab. w ist dabei ein frei wählbarer Parameter[10]. Die Pheromonmenge der r't-besten Ameise wird dann zusätzlich entsprechend ihrem Rang mit $(w-r)$ gewichtet. Insgesamt ergibt sich dann für das Pheromon-Update

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \sum_{r=1}^{w-1}(w-r)\Delta\tau_{ij}^{r} + w\Delta\tau_{ij}^{\text{elitist}}. \tag{6.7}$$

Die Grund-Pheromonmenge der r't-besten Ameise $\Delta\tau_{ij}^{r}$ sowie der *Elitist Ant* $\Delta\tau_{ij}^{\text{elitist}}$ werden entsprechend Gleichung 6.3 bestimmt.

Bullnheimer *et al.* (1997a) zeigen bei Anwendung auf das TSP eine Überlegenheit des *Rank-Based Ant System* gegenüber dem ursprünglichen *Ant System* und dem *Elitist Ant System*. Allerdings ist bei großen Probleminstanzen ($n \geq 96$) keine der dort untersuchten Methoden fähig, die optimale Lösung zu finden.

6.2.4. Ant Colony System

Das *Ant Colony System* (Gambardella & Dorigo, 1996; Dorigo & Gambardella, 1997) stellt eine noch stärkere Abstraktion gegenüber dem Vorbild der realen Ameisenkolonie dar als die anderen hier behandelten Ameisenalgorithmen. Gegenüber dem ursprünglichen *Ant System* gibt es drei wesentliche Änderungen:

1. Für die Wahl der nächsten zu besuchenden Stadt wird ein aggressiveres Auswahlkriterium angewendet.
2. Nur eine einzige Ameise, die beste bisher gefundene Lösung, lagert Pheromon ab.
3. Zusätzlich findet ein lokales Pheromon-Update statt, bei dem die Ameisen der Kolonie im Laufen Pheromon entfernen.

[8] Beim TSP der Länge der gefundenen Route
[9] engl. *rank*
[10] Für das TSP empfehlen Dorigo & Stützle (2004) $w = 6$.

Auf diese Punkte wird nun im Einzelnen genauer eingegangen.

Im *Ant Colony System* erfolgt die Wahl der nächsten zu besuchenden Stadt nicht mehr rein probabilistisch entsprechend Gleichung 6.1. Stattdessen wird die Wahl von kurzen Kanten mit hohen Pheromonmengen durch das Einführen einer Greedy-Regel zusätzlich begünstigt. Eine Ameise in Stadt i wählt mit einer Wahrscheinlichkeit q_0 die Stadt j mit dem höchsten Produkt aus heuristischer Information η_{ij} und Pheromonmenge τ_{ij}. Ansonsten wendet sie Gleichung 6.1 an[11]. Konkret folgt die Wahl der nächsten Stadt j

$$j = \begin{cases} \text{argmax}_{l \in N_i^k} \left\{ [\tau_{il}]^\alpha \cdot [\eta_{il}]^\beta \right\} & \text{, wenn } q \leq q_0 \\ J & \text{, sonst.} \end{cases} \tag{6.8}$$

q ist dabei eine Zufallsvariable $0 \leq q \leq 1$ und J eine Stadt, die nach Gleichung 6.1 gewählt wird.

Diese pseudozufällig-proportionale Wegewahl hat gegenüber der reinen Roulette-Wahl des *Ant System* den Vorteil, dass die Ausnutzung (*exploitation*) der bisher gefundenen Lösungen gegenüber der Erkundung (*exploration*) des gesamten Lösungsraumes einerseits stärker gewichtet wird (vgl. Abschnitt 2.4), andererseits diese relative Gewichtung von *exploitation* und *exploration* über den Parameter q_0 individuell und problemspezifisch gesetzt werden kann.

Die beste bisher gefundene Lösung, und damit die *exploitation*, wird dadurch zusätzlich noch stärker gewichtet, dass die Regel des Pheromon-Updates gegenüber dem *Elitist Ant System* und dem *Rank-Based Ant System* noch verschärft wird. Nur noch die *Elitist Ant*, also die beste bisher gefundene Lösung, lagert Pheromon ab. Allerdings erfolgt auch die Pheromonverdunstung nur auf den Kanten, die zur bisher besten Lösung gehören. Dieses globale Pheromon-Update folgt der Gleichung

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho \Delta \tau_{ij}^{\text{elitist}} \qquad , \forall (i,j) \in T^{\text{elitist}}. \tag{6.9}$$

Dabei ist T^{elitist} die Route der *Elitist Ant* und $\Delta\tau_{ij}^{\text{elitist}}$ berechnet sich wie im *Elitist Ant System* entsprechend Gleichung 6.3. Die Multiplikation von $\Delta\tau_{ij}^{elitist}$ mit dem Faktor ρ führt dazu, dass die neue Pheromonmenge auf der Kante (i,j) einem gewichteten Durchschnitt aus der alten Pheromonmenge und der neu abgelagerten Menge entspricht (Dorigo & Stützle, 2004). Dies verhindert eine übermäßige Gewichtung der Kanten der Elitist-Lösung.

Dorigo & Gambardella (1997) beschreiben auch die Möglichkeit, in Gleichung 6.9 statt der Elitist-Lösung die beste Lösung der jeweiligen Iteration zu verwenden. Für kleine Probleminstanzen verhält sich der Algorithmus dann ähnlich, für größere Probleminstanzen ($n > 100$) zeigt sich aber die Verwendung der Elitist-Lösung als überlegen.

Die dritte Änderung des *Ant Colony Systems* gegenüber dem *Elitist Ant System* sorgt dafür, dass die starke Gewichtung der *exploitation* nicht dazu führt, dass der

[11] In Dorigo & Stützle (2004) wird für das TSP $q_0 = 0.9$ empfohlen.

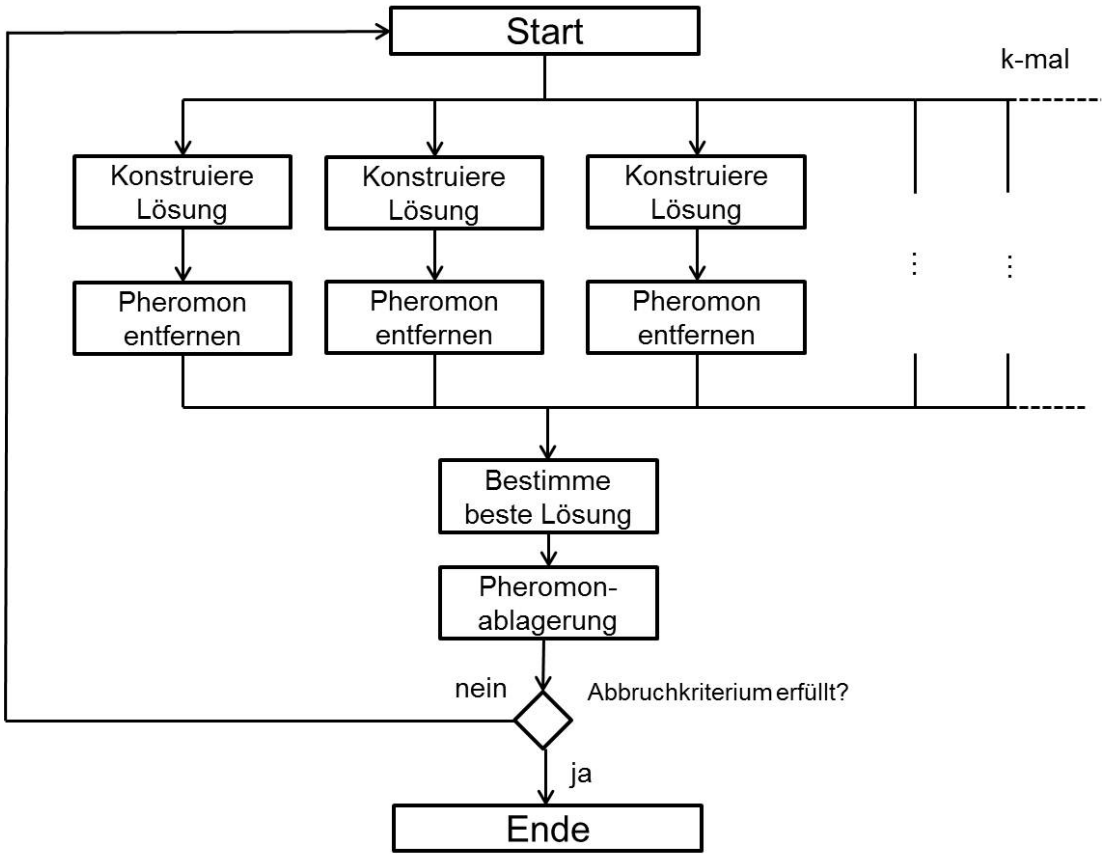

Abbildung 6.4.: Ablauf des Ant Colony Systems

Algorithmus in einem lokalen Optimum stagniert. Dazu wird die Pheromonmenge auf einer Kante (i, j) noch während der Tourkonstruktion entsprechend

$$\tau_{ij} \leftarrow (1 - \xi)\tau_{ij} + \xi\tau_0 \tag{6.10}$$

verringert, sobald eine Ameise diese Kante wählt. τ_0 entspricht dabei der Ausgangsmenge von Pheromon auf allen Kanten und wird wie in Abschnitt 6.2.1 beschrieben bestimmt. ξ ist ein Parameter $0 < \xi < 1$. Normalerweise wird $\xi = \rho$ gesetzt. Mit diesem lokalen Pheromon-Update werden die Ameisen dazu gebracht, Kanten „auszupobieren“, die bisher noch kaum besucht waren und so neue Regionen des Lösungsraumes zu erkunden. Der gesamte Ablauf des *Ant Colony Systems* ist in Abbildung 6.4 zusammengefasst dargestellt.

Einen großen Vorteil des *Ant Colony Systems* gegenüber anderen Dialekten der Ameisenalgorithmen ist die sehr geringe Anzahl an Ameisen m, die hier benötigt wird. In Dorigo & Gambardella (1997) wird der ideale Wert für m halb empirisch wie folgt hergeleitet:

Man setzt voraus, dass der durchschnittliche Pheromonlevel auf den Kanten der besten Tour einer Iteration vor dem globalen Pheromon-Update das ϕ_1-fache und nach diesem das ϕ_2-fache der Pheromongrundmenge τ_0 ist und die idealen Werte

für ϕ_1 und ϕ_2 bekannt sind. Durch das lokale Pheromon-Update ändert sich die durchschnittliche Pheromonmenge T_z auf einer Kante der besten bisher bekannten Tour nach der z-ten Ameise, die sie quert (mit $\xi = \rho$), auf

$$T_z = T_{z-1}(1-\rho) + \tau_0\rho \tag{6.11}$$

bzw. in geschlossener Form

$$T_z = T_0(1-\rho)^z - \tau_0(1-\rho)^z + \tau_0 \tag{6.12}$$

T_0 ist dabei die durchschnittliche Pheromonmenge auf den Kanten der besten bisher bekannten Tour zu Beginn des Iterationsschrittes.

Entsprechend der vorausgehenden Überlegung ist $T_0 = \phi_2\tau_0$. Nachdem alle Ameisen ihren Weg beendet haben aber vor dem globalen Update ist der Wert für $T_z = \phi_1\tau_0$. Zusammengefasst ergibt sich also

$$\phi_1 = \phi_2(1-\rho)^z - (1-\rho)^z + 1 \tag{6.13}$$

Geht man davon aus, dass die Kanten der besten bekannten Lösung mit einer Wahrscheinlichkeit von q_0 gewählt werden[12], dann entspricht die Anzahl z der Ameisen auf ihren Kanten mq_0. Die ideale Anzahl bestimmt sich dann zu

$$m_{ideal} = \frac{\log(\phi_1 - 1) - \log(\phi_2 - 1)}{q_0 \log(1-\rho)} \tag{6.14}$$

Dorigo & Gambardella (1997) ermittelten empirisch, dass der Algorithmus für ein Verhältnis $\frac{\phi_1-1}{\phi_2-1} = 0.4$ die besten Ergebnisse lieferte. Genauere Abhängigkeiten, beispielsweise von der Problemgröße, sind bis heute nicht bekannt. Bei $q_0 = 0.9$ ergibt sich für $\rho = 0.1$ eine ideale Koloniegröße von $m = 10$, für $\rho = 0.2$ ergibt sich $m = 5$ und für $\rho = 0.4$ sollte die Kolonie nur noch zwei Ameisen umfassen.

In Dorigo & Gambardella (1997) zeigen die Autoren die gute Leistung des ACS im Vergleich zu anderen Heuristiken (unter anderem Genetische Algorithmen und Simulated Annealing) auf kleinen (50 bis 100 Städte) Instanzen des TSP. Dabei liefert ACS von den untersuchten Algorithmen die besten Lösungen und muss dazu weniger Lösungen auswerten. In Dorigo & Stützle (2004) findet sich ein Vergleich von ACS mit anderen Dialekten von Ameisenalgorithmen in der Anwendung auf zwei große TSP-Instanzen (d198 und rat783 aus der TSPLIB[13], die Zahl im Namen der Instanz bezeichnet die Anzahl an Städten in der Probleminstanz). Bei beiden zeigt sich ACS den anderen hier beschriebenen Dialekten überlegen, indem es bessere Lösungen in einer kürzeren Rechenzeit findet.

[12] In Wirklichkeit liegt die Wahrscheinlichkeit geringfügig höher
[13] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

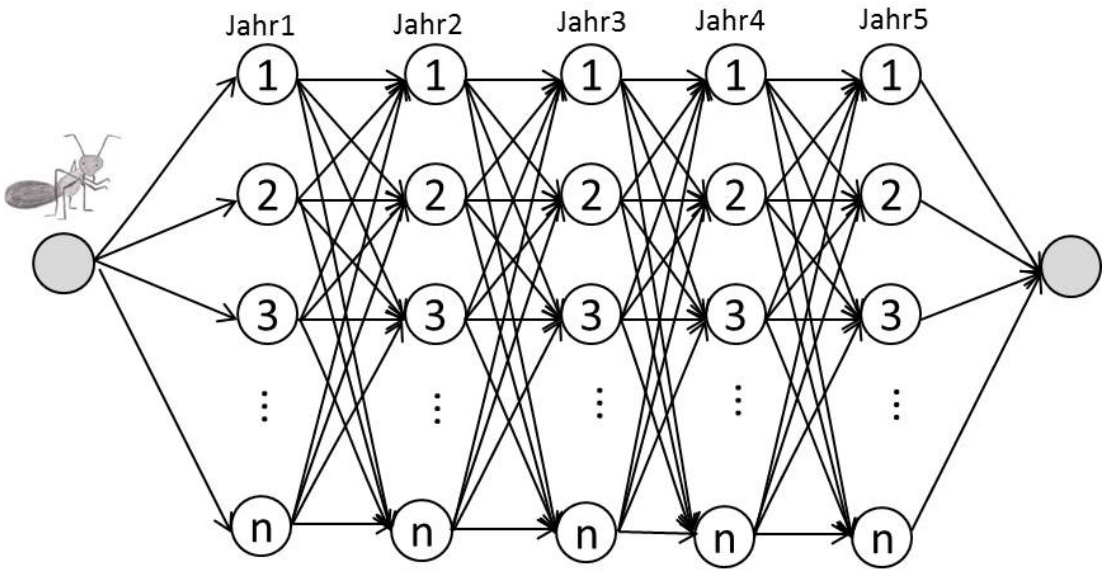

Abbildung 6.5.: Konstruktionsgraph für das Instandsetzungsproblem. Der Graph hat y Schichten für die betrachteten Jahre und n Knoten pro Schicht für die betrachteten Bauwerke

6.3. Ameisenalgorithmen für das Lebensdauermanagement

6.3.1. Formulierung des Problems

Zur Behandlung mit Ameisenalgorithmen muss für ein Problem zunächst ein Konstruktionsgraph formuliert werden (vgl. Gutjahr (2000),Cordón García *et al.* (2002) oder Dorigo & Stützle (2004)). Dabei handelt es sich um einen (gewichteten) Graph, bei dem die (Teil-)Lösungen des Problems als Routen durch diesen Graph beschrieben werden können.

Für die Optimierung von Instandsetzungszeitplänen bietet sich die Formulierung als geschichteten Graph mit y Schichten und n Knoten pro Schicht an. y ist dabei die Anzahl der betrachteten Jahre und n die Anzahl der betrachteten Bauwerke (Abbildung 6.5).

Der Suchraum kann dadurch begrenzt werden, dass Knoten, für die die Reparaturfrist überschritten wurde (z. B. Knoten $3j$, wenn die Frist für Bauwerk j bereits im Jahr 2 abläuft), aus der Suche ausgeschlossen werden, bzw. die Wahrscheinlichkeit ihrer Wahl konstant gleich Null gesetzt wird. Anders als andere Begrenzungen des Suchraumes hat dies keine negativen Auswirkung auf die Suche, sondern kann diese sogar verkürzen, da es die Suche in sehr ungünstigen Regionen verhindert: Eine Lösung ist ungültig, egal ob ein Bauwerk nach Überschreitung seiner Frist terminiert oder aber überhaupt nicht innerhalb des Be-

trachtungszeitraumes berücksichtigt wird (vgl. Abschnitt 3.3). Andererseits kann es sein, dass ausgerechnet die Kombination dieses Bauwerks mit einigen anderen in einem Jahr nach der Fristüberschreitung sich besonders günstig auf die Zielfunktion auswirkt. Die Lösungen in unmittelbarer Nachbarschaft dieser Lösung sind zum größten Teil also ebenfalls ungültig, wenn das betroffene Bauwerk weiterhin in dem zu späten Jahr terminiert wird. Eine Verschiebung der Reparatur des betroffenen Bauwerks nach vorne könnte dagegen gravierende Auswirkungen auf die Zielfunktion haben. Bei ungültigen Lösungen, bei denen diese Knoten nicht zugelassen sind, ist dagegen ein wesentlich größerer Teil der Nachbarschaft gültig und die Zielfunktionswerte eines ungültigen und eines gültigen Nachbarn unterscheiden sich im Mittel nicht so stark.

In diesem Zusammenhang ist noch mal eine genauere Definition des Nachbarschaftsbegriffes notwendig. Wie bereits in Abschnitt 2.2 erwähnt, gibt es bei kombinatorischen Optimierungsproblemen keine eindeutige Definition für die Nachbarschaft einer Lösung. Welche Lösungen „nahe“ beieinander liegen hängt dabei auch davon ab, wie die Lösungen aufgebaut sind und wie sie konstruiert werden. Eine Lösung des Instandsetzungsproblems besteht aus einer „Route“ durch den Graph, bei der in jeder Schicht l Knoten (Bauwerke)[14] ausgewählt wurden. Dabei ist es wesentlich wahrscheinlicher, dass eine Ameise bei der Lösungskonstruktion in einem einzigen Knoten von einer stark mit Pheromon markierten Route x abweicht, und so die Lösung x' erzeugt, als dass sie für alle l Einträge eine Schicht andere Knoten wählt. Letzteres wäre dann die Lösung x''. Damit liegen die Lösungen x und x' wesentlich näher zusammen als die Lösungen x und x''. Wenn im Folgenden von Nachbarschaft die Rede ist, so ist eine Nachbarschaft in diesem Sinne gemeint. Eine genauere Ausführung zu diesem Punkt folgt in Abschnitt 6.3.5.

Die Auswahl einer festgelegten Anzahl an Knoten pro Schicht kann auf zwei Arten erreicht werden: Entweder bewegt sich eine Ameise so lange innerhalb einer Schicht, bis sie die festgesetzte Anzahl an Knoten in dieser Schicht besucht hat. Erst dann darf sie zur nächsten Schicht überwechseln.

Wesentlich eleganter ist es aber, keine einzelnen Ameisen zur Lösungskonstruktion auszuschicken, sondern „Teams“ von Ameisen. Dieser Ansatz ist inspiriert von einer Idee aus Lee (2009). Dort vertreten die einzelnen Ameisen eines Teams jeweils eine Gruppe von Arbeitern, die sich von einer Straßenbaustelle zur nächsten weiterbewegt. Dadurch können die Terminpläne für alle Arbeitergruppen gleichzeitig erstellt und optimiert werden. Für das in dieser Arbeit behandelte Problem fehlt eine derart anschauliche Entsprechung für die Ameisenteams. Dennoch ist ihre Verwendung hier sinnvoll, da so ein Agent aus der Population (ein Team bestehend aus l Ameisen) in einem Schritt genau l Knoten aus einer Schicht auswählen und schon im nächsten Schritt zur nächsten Schicht übergehen kann.

[14] pro Jahr dürfen genau l Bauwerke instand gesetzt werden

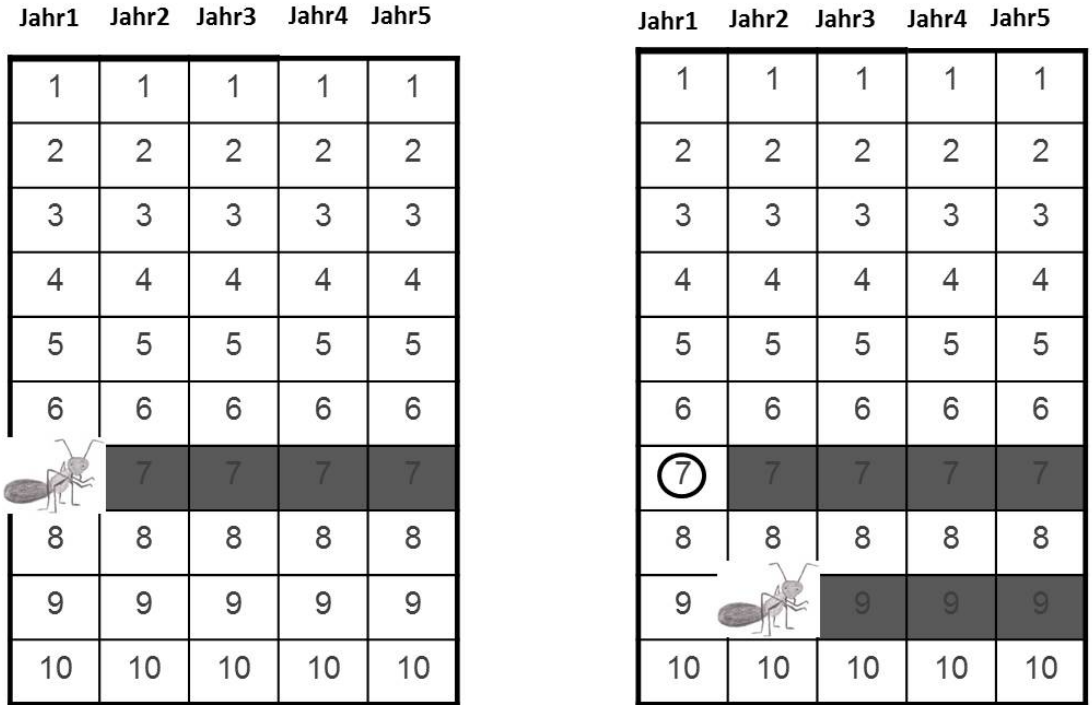

Abbildung 6.6.: Vereinfachter Konstruktionsgraph für $n = 10$, $y = 5$ und $l = 1$ in den ersten zwei Schritten der Lösungskonstruktion. Für das erste Jahr wird Bauwerk 7 ausgewählt, damit ist es für alle weiteren Jahre tabu. Durch die Wahl von Bauwerk 9 für das zweite Jahr kann im dritten Jahr nur noch aus den verbliebenen acht Bauwerken gewählt werden.

Die Entscheidungen einer Ameise aus einem Team sind dabei allen Ameisen des Teams bekannt. Das bedeutet, ein Bauwerk, das in einem Jahr von einer Ameise aus Team k gewählt wurde, ist für alle Ameisen dieses Teams für alle weiteren Jahre tabu (vgl. Abbildung 6.6). Die Ameisen aller weiteren Teams sind davon nicht betroffen.

Da es für die Güte eines Instandsetzungszeitplans darauf ankommt, welche Bauwerke gleichzeitig in welchem Jahr terminiert wurden, nicht aber, welches Bauwerk direkt davor oder direkt danach instand gesetzt wurde, macht es wenig Sinn, die Pheromonablagerungen auf die Kanten des Problemgraphen zu beziehen. Stattdessen erfolgt die Pheromonablagerung hier auf den Knoten des Graphen. Die Pheromonmatrix hat damit die gleiche Struktur wie der Konstruktionsgraph selber: Es handelt sich um eine $n \times y$-Matrix. τ_{ij} bedeutet dann nicht mehr, wie in der ursprünglichen Formulierung, die Pheromonmenge auf der Verbindung zwischen den Knoten i und j, sondern das Pheromon auf dem Knoten j in Spalte i, also für die Attraktivität der Wahl Bauwerk j im Jahr i instand zu setzen. Eine Ameise, die sich auf einem beliebigen Knoten in der Spalte $i - 1$ befindet, wählt dann als nächsten Schritt einen beliebigen (erlaubten) Knoten in Spalte i entsprechend Gleichung 6.1 (bzw. Gleichung 6.8 für das *Ant Colony System*).

6.3.2. Heuristische Information η

Für die Entscheidungsfindung ist ein weiterer Punkt zu beachten: Zur Definition der heuristischen Information η ist es nötig zu wissen, welchen Effekt eine Entscheidung auf die Lösungsgüte hat. Dies lässt sich bei dem hier beschriebenen Problem allerdings für die Auswahl eines einzelnen Knotens nicht ermitteln, da die Lösungsgüte hier immer von der Kombination der Bauwerke in einem Jahr abhängt, nicht aber davon, ob ein einzelnes Bauwerk ein Jahr früher oder später terminiert wird. Somit kann aus Sicht der Optimierungsgröße „Einfluss auf den Verkehr“ keine sinnvolle Heuristik gefunden werden, die die Wahl eines bestimmten Knotens unterstützt. Betrachtet man rein den Verkehr, müssten also anfangs alle Knoten die gleiche Attraktivität besitzen. Der Parameter η verliert damit zunächst seine Bedeutung. Das könnte problematisch sein, da η nach Birattari *et al.* (2002) vor allem auch deswegen benötigt wird, weil der Konstruktionsggraph nicht alle lösungsrelevanten Informationen abbilden kann[15].

Es bietet sich allerdings an, diesen so frei gewordenen Parameter anders zu nutzen, um den Ameisen bei ihrer Suche zusätzliches Wissen mitzugeben. Wenn dieses Wissen schon nicht dabei helfen kann, Informationen über den Einfluss einer Wahl auf die Lösungsgüte zu liefern, so kann es doch dazu dienen, die Ameisen bevorzugt in gültige Regionen zu steuern. Zur Erinnerung: Ein Instandsetzungszeitplan ist, lässt man die monetären Grenzen außer Acht, dann gültig, wenn jede Brücke vor ihrem individuellen Fristablauf terminiert wurde (vgl. Abschnitt 3.3.2). Bauwerke mit einer niedrigen Frist sollten also möglichst früh in der Terminplankonstruktion berücksichtigt werden, das heißt für die Ameisen attraktiver sein, Bauwerke mit einer sehr hohen Frist sollten dagegen eher unattraktiv sein. Basierend auf diesen Überlegungen wird der Wert η_{ij} umgekehrt proportional dazu gewählt, wie weit Bauwerk j im Jahr i noch von seinem Fristablauf entfernt ist.

Damit bei der Entscheidung nach Gleichung 6.1 weder η noch τ einen ungeplant großen Einfluss haben, muss die Größenordnung dieser beiden Parameter entsprechend angepasst werden: Die Instandsetzungsfrist für die einzelnen Brücken hat einen Wert zwischen 1 und ca. 20 Jahren, der Einfluss eines Zeitplans auf den Verkehr dagegen, gemessen in Fahrzeugsekunden im gestörten Netz, bewegt sich in der Größenordnung von 10^8. Würde man für η nun einfach $1/dl$ setzen[16], $\Delta\tau$ dagegen jeweils nach Gleichung 6.3 berechnen[17], hätten unterschiedliche Instandsetzungsfristen auch in späteren Iterationen einen wesentlich stärkeren Einfluss auf die Routenwahl der Ameisen als die Pheromonspuren. Dies würde dem Prin-

[15] Eine solche vollständige Abbildung wäre nur über einen Statusgraph möglich, in dem die Knoten für Teillösungen des Problems stehen. Einen solchen Graph verwendet das *Ant Programming* (Birattari *et al.*, 2002)

[16] mit dl dem Abstand zwischen dem betrachteten Jahr und dem Fristende der betrachteten Brücke

[17] mit C den Fahrzeugsekunden im durch den betrachteten Instandsetzungszeitplan beeinträchtigten Netz

zip des Algorithmus zuwieder laufen. Daher muss τ zunächst um einen Faktor 10^8 vergrößert werden, bevor es in die Gleichung 6.1 eingesetzt werden kann.

Diese Definition von η sorgt dafür, dass Bauwerke mit niedriger Instandsetzungsfrist bevorzugt ausgewählt werden. Ein Terminplan ist aber nur dann gültig, wenn auch die Kosten in den einzelnen Jahren innerhalb des zulässigen Bereichs liegen. Diese Randbedingung wird bei obiger Definition von η nicht berücksichtigt. Für *Ant System*, *Elitist Ant System* und *Rank-based Ant System* macht dies kaum einen Unterschied, da die Wahl der Ameisen dort recht frei geschieht und so mit großer Wahscheinlichkeit auch gute Lösungen gefunden werden, die beide Randbedingungen erfüllen. Beim *Ant Colony System* dagegen ist die Suche durch das strengere Entscheidungskriterium und die geringere Anzahl an Ameisen stark eingeschränkt. Mit einem wie beschrieben definierten η führt dies dazu, dass sehr leicht Lösungen gefunden werden, die die Sicherheitsrandbedingung erfüllen. Allerdings ist es sehr viel schwerer solche Lösungen zu finden, die dies auch mit der Kostenrandbedingung tun[18].

Diesem Problem kann man dadurch begegnen, dass in die Bestimmung von η auch die Kostenrandbedingung eingeht. Da diese Randbedingung aber nicht von den einzelnen Bauwerken abhängt sondern nur von deren Kombination kann dies nicht global für die gesamte Optimierung geschehen. Stattdessen muss η in jeder Iteration für jede Ameise für jedes Bauwerk für jedes Jahr des Terminplanes dynamisch angepasst werden.

Dazu werden nach der Wahl eines Bauwerks bei der Konstruktion eines Terminplanes zunächst für das Jahr i, für das die Wahl getroffen wurde, die nun erwarteten Kosten bestimmt. Diese berechnen sich aus den durchschnittlichen Kosten aller bisher für i gewählten Bauwerke multipliziert mit l. Liegen diese Kosten innerhalb der vorgegebenen Schranken, so wird η weiterhin nur über die Instandsetzungsfrist berechnet.

Beispiel: *Die Kosten pro Jahr sollen zwischen 2 und 3 Mio Euro. liegen. Als erstes Bauwerk für das Jahr 1 des Terminplanes wird dasjenige mit der Nummer 59 gewählt. Die Instandsetzungskosten für dieses Bauwerk betragen 212 600 Euro. Pro Jahr sollen 10 Bauwerke instand gesetzt werden. Die geschätzten Kosten für das erste Jahr betragen also 2.12 Mio. Euro. Damit liegen sie innerhalb des erlaubten Bereichs. Zur Wahl des zweiten Bauwerks für das erste Jahr wird η also für alle Bauwerke auf* $10^{-8}/dl$ *gesetzt.*

Liegen die erwarteten Kosten für das Jahr i oberhalb der oberen Budget-Schranke, so müssen nun bevorzugt solche Brücken gewählt werden, deren Kosten niedrig

[18] Man stelle sich eine Probleminstanz vor, bei der alle Bauwerke mit niedriger Instandsetzungsfrist sehr hohe Kosten haben, Bauwerke mit niedrigen Kosten haben dagegen sehr hohe Instandsetzungsfristen: Eine sehr „gierige“ Auswahl der Bauwerke rein nach den Instandsetzungsfristen würde nun dazu führen, dass für die ersten Jahre des Terminplanes fast ausschließlich teure Bauwerke mit niedriger Instandsetzungsfrist gewählt würden. Damit ist dann zwar die Sicherheitsrandbedingung erfüllt, nicht aber die Kostenrandbedingung

sind. η wird nun also für alle Bauwerke für das Jahr j so modifiziert, dass es für Bauwerke mit niedrigen Kosten höher ist:

$$\eta_{i,j} = \frac{s_{dl}}{dl_{i,j}} * \frac{s_{\text{cost}}}{c_i} \tag{6.15}$$

wobei s_{dl} bzw. s_{cost} geeigenete Skalierungsfaktoren für die Instandsetzungsfrist bzw. die Kosten sind und c_i die Kosten für eine Instandsetzung des Bauwerks i beschreibt.

Beispiel, fortgeführt: *Als zweites Bauwerk für das Jahr 1 wird nun dasjenige mit der Nummer 61 ausgewählt. Die Instandsetzungskosten für dieses Bauwerk liegen bei 395 700 Euro. Die geschätzten Kosten für das Jahr 1 betragen nun also* $(212600 + 395700)/2 * 10 = 3.04$ *Mio. Euro und liegen damit oberhalb der oberen Budgetgrenze.* η *wird nun also für alle Bauwerke entsprechend Gleichung 6.15 angepasst. Für das Bauwerk 67 mit einer Instandsetzungsfrist von 2 und Instandsetzungskosten in der Höhe von 152 800 Euro hatte* η_{67} *zuvor die Größe* $2 * 10^{-8}$*, nun (mit* $s_{cost} = 10^5$*) berechnet es sich auf* $2.35 * 10^{-8}$*. Für Bauwerk 58 dagegen, ebenfalls mit Instandsetzungsfrist von 2 aber Instandsetzungskosten von 388.700 Euro, beträgt* η_{58} *nur noch* $1.29 * 10^{-8}$*.*

Liegen die erwarteten Kosten dagegen unterhalb der unteren Budget-Schranke, dann werden die Bauwerke mit hohen Instandsetzungskosten attraktiver gemacht durch:

$$\eta_{i,j} = \frac{s_{dl}}{dl_{i,j}} * \frac{c_i}{s_{\text{cost}}} \tag{6.16}$$

Beispiel, fortgeführt: *Als zweites Bauwerk für das Jahr 1 wird statt Bauwerk Nummer 61 Bauwerk Nummer 35 ausgewählt. Die Instandsetzungskosten für dieses Bauwerk liegen bei 112 400 Euro. Die geschätzten Kosten für das Jahr 1 betragen nun also* $(212600+112400)/2*10 = 1.63$ *Mio. Euro und liegen damit unterhalb der unteren Budgetgrenze.* η *wird nun also für alle Bauwerke entsprechend Gleichung 6.16 angepasst. Für das Bauwerk 67 hat* η *nun den Wert* $7.64 * 10^{-8}$*, für die Bauwerk 58 dagegen den Wert* $1.94 * 10^{-7}$*.*

6.3.3. Dynamische Anpassung von β

Durch die Umdeutung von η ergibt sich aber ein neues Problem: Durch die bevorzugte Wahl der Bauwerke mit naheliegender Frist wird ein Teil des Lösungsraumes, nämlich der in Nähe der Greedy Lösung, übermäßig beorzugt. Der Algorithmus läuft damit häufig in ein lokales Optimum innerhalb dieses Bereichs.

Eine mögliche Abhilfe ist es, β über die Iterationen schrittweise zu reduzieren, wie in Merkle *et al.* (2002) beschrieben. Damit bleiben zumindest die gültigen Lösungen der ersten Iterationen erhalten, dem Algorithmus wird aber die Freiheit

gelassen, den weiteren Lösungsraum abzusuchen. Eine mögliche Formulierung für ein dynamisches β ist

$$\beta(it) = \beta_0 - it * \frac{\beta_0}{n_{\text{iterations}}} \tag{6.17}$$

Dabei ist β_0 der Startwert für β, it die betreffende Iteration und $n_{\text{iterations}}$ die Gesamtzahl der Iterationen.

Mit diesen in den Abschnitten 6.3.1 bis 6.3.3 beschriebenen Änderungen wurden das *Ant System*, das *Elitist Ant System*, das *Rank-based Ant System* sowie das *Ant Colony System* für die Optimierung der Instandsetzungszeitpläne implementiert.

6.3.4. Behandlung ungültiger Lösungen

In der ursprünglichen Beschreibung von Ameisenalgorithmen ist es nicht vorgesehen, dass ungültige Lösungen auftreten. Es wird dabei davon ausgegangen, dass sich schon während der Lösungskonstruktion vermeiden lässt solche Lösungen zu erstellen, beispielsweise dadurch, dass im TSP bereits besuchte Städte tabu sind. Dies ist allerdings für das Instandsetzungsproblem nicht möglich, da sich oft erst während der Konstruktion entscheidet, ob eine Lösung gültig ist oder nicht. So ist zum Beispiel erst nach der Wahl aller Knoten der Schicht j klar, ob ein kritisches Bauwerk mit der Frist j rechtzeitig in den Terminplan aufgenommen wurde und die Lösung damit gültig ist oder nicht. Auch Entscheidungen dazu, ob die Kostenrandbedingung erfüllt ist, sind jeweils erst nach der Wahl aller Bauwerke für ein Jahr möglich.

Aus diesem Grund müssen Regeln gefunden werden, wie die so entstandenen ungültigen Lösungen zu behandeln sind. Die in Abschnitt 6.3.2 beschriebene Umdeutung des Parameters η hilft dabei schon, die Suche verstärkt in gültige Regionen des Lösungsraumes zu lenken. Diese Maßnahme allein ist allerdings nicht ausreichend

Für das *Ant System* zeigen sich beim Vorhandensein ungültiger Lösungen zusätzlich zu dem unter 6.2.2 beschriebenen Problem des Verlusts guter Lösungen noch weitere Probleme: Da in die Pheromonablagerung zunächst nicht eingeht, ob eine Lösung gültig ist oder nicht, besteht die Gefahr, dass der Algorithmus gegen eine ungültige Lösung konvergiert. Wie bei den Genetischen Algorithmen (vgl. Abschnitt 5.2.3) gibt es auch hier die drei Möglichkeiten:

- die ungültigen Lösungen nicht beachten, also auf ihnen kein Pheromon ablagern,
- die ungültigen Lösungen mit einer Strafe versehen, oder
- versuchen die ungültigen Lösungen zu reparieren.

Die ersten beiden Möglichkeiten zeigen sich durch den nicht zusammenhängenden und stark zerfaserten gültigen Lösungsraum problematisch: Eine gute gültige

Lösung kann sich in unmittelbarer Nachbarschaft von einer ungültigen Lösung befinden, eventuell sogar den gleichen Wert für die Zielfunktion haben. Werden nun ungültige Lösungen grundsätzlich nicht beachtet oder mit einer zu hohen Strafe versehen, so gehen diese gültigen Lösungen in ihrer Nachbarschaft ebenfalls für den Algorithmus verloren, bzw. werden nur mit einer äußerst geringen Wahrscheinlichkeit gefunden. Die Suche beschränkt sich dann eher auf einzelne Inseln von gültigen Lösungen (vgl. Abbildung 5.10). Einzig das Einführen einer Reparaturfunktion kann hier Abhilfe bieten.

Eine Reparatur erfolgt ähnlich dem entsprechenden Mechanismus bei den Genetischen Algorithmen (Abschnitt 5.3.3.4). Dazu wird eine Liste der Bauwerke vorgehalten, die „kritisch“ sind, deren Frist also innerhalb des Betrachtungszeitraums abläuft. Soll nun eine ungültige Lösung repariert werden, so wird für jedes dieser kritischen Bauwerke überprüft, ob es in dem Terminplan berücksichtigt wird[19]. Ist dies nicht der Fall, so wird versucht das Bauwerk in den Plan einzufügen: Ausgehend von dem Jahr, in dem die Frist abläuft, nach vorne gehend wird ein Bauwerk gesucht, dessen Frist außerhalb des Betrachtungszeitraumes liegt. Wird eine solche gefunden, so wird es aus dem Plan entfernt und durch das kritische Bauwerk ersetzt (vgl. Abbildung 6.8). Dann wird die Liste kritischer Bauwerke weiter abgearbeitet. Gelingt es nicht, das kritische Bauwerk gegen ein Bauwerk mit hoher Frist zu tauschen, dann bleibt die Lösung auch nach der Reparatur ungültig. Allerdings weist sie eventuell durch die geglückten Tauschaktionen nun eine geringere Menge an Randbedingungsverletzungen auf.

Als ebenfalls gut geeignet zum Steuern der Suche weg von ungültigen Regionen, ohne die ungültigen Lösungen zu stark zu benachteiligen, zeigt sich die Verwendung der *Elitist Ant* (vgl. 6.2.2). Nach obiger Einteilung handelt es sich dabei um eine Art Strafe für die ungültigen Lösungen. Allerdings wird diese Strafe nicht direkt sondern indirekt vergeben, indem nur gültige Lösungen als *Elitist Ant* zugelassen werden. So lange noch keine gültige Lösung gefunden wurde, gibt es auch keine *Elitist Ant*, das heißt, so lange erfolgt die Suche wie im *Ant System*. Eine existierende *Elitist Ant* kann nur dann durch eine bessere (im Hinblick auf die Zielfunktion) Lösung ersetzt werden, wenn diese auch gültig ist. Alle anderen Lösungen lagern ihr Pheromon entsprechend Gleichung 6.6 ab, gleichgültig ob sie gültige oder ungültige Lösungen beschreiben. Dies soll dafür sorgen, dass auch weiterhin die Nachbarschaft von vielversprechenden ungültigen Lösungen abgesucht wird, in der Hoffnung dort auch auf gute gültige Lösungen zu stoßen.

Ähnlich lässt sich beim *Rank-based Ant System* verfahren. Da auch hier eine *Elitist Ant* vorgehalten wird, die in jeder Iteration, solange keine bessere Lösung gefunden wurde, Phermon ablagert, kann auch hier die Regel eingeführt werden, dass nur gültige Lösungen diese besetzen dürfen. In die „elitäre Gruppe“ der Größe w, die Pheromon ablagert, werden Lösungen dagegen, unabhängig davon ob sie gültig oder ungültig sind, aufgenommen. In Extremfällen, vor allem in den

[19] anders als bei den Genetischen Algorithmen kann ein kritisches Bauwerk nur vor Ablauf ihrer Frist oder aber überhaupt nicht Teil des Terminplanes sein (vgl. Abschnitt 6.3.1)

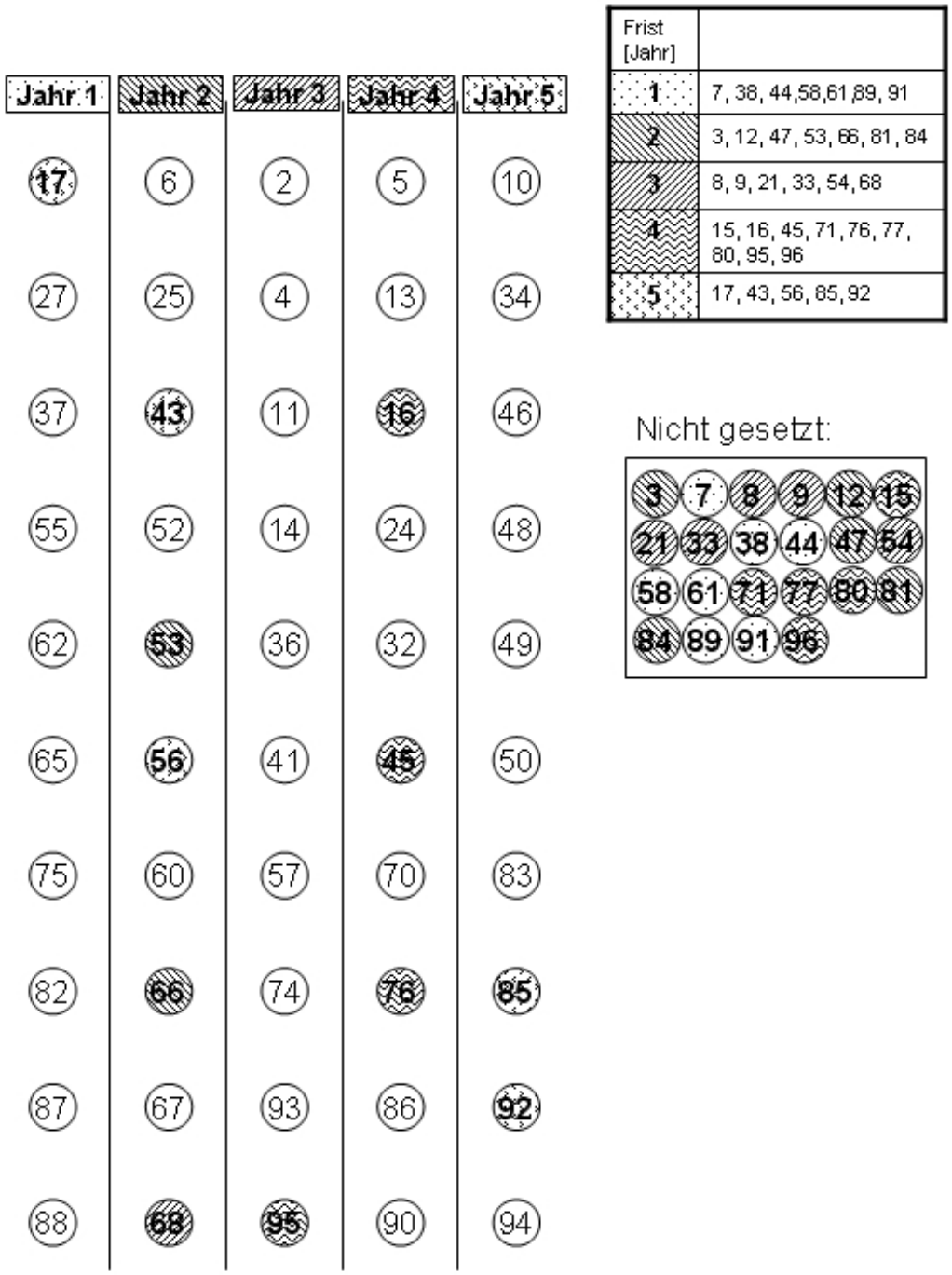

Abbildung 6.7.: Ungültige Lösung: 22 kritische Bauwerke sind nicht Teil des Terminplans.

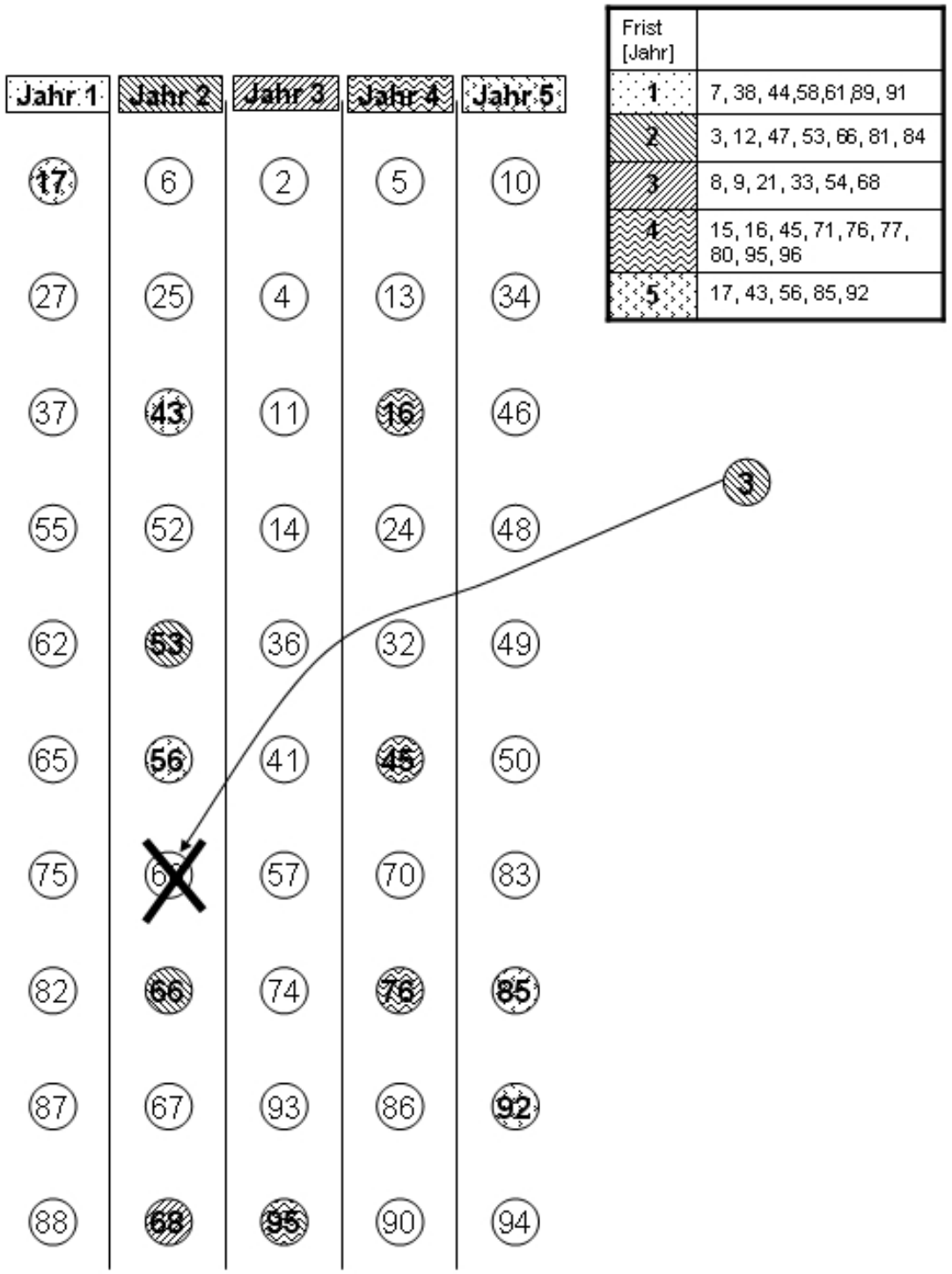

Abbildung 6.8.: Erster Schritt der Reparatur der Lösung aus Abbildung 6.7. Die Frist für Bauwerk 3 liegt im Jahr 2, das Bauwerk 3 ist aber bisher nicht Teil des Terminplans. Im Jahr 2 gibt es mehrere Bauwerke deren Frist außerhalb des Betrachtungszeitraumes liegt. Eines davon, Bauwerk 60, wird durch das Bauwerk 3 ersetzt. Im nächsten Schritt erfolgt das gleiche für Bauwerk 7, usw.

ersten Iterationen eines Durchlaufs, führt dies dazu, dass eine Lösung zur *Elitist Ant* wird und als solche Pheromon ablagert, die gar nicht zu den besten $(w-1)$ Lösungen der Iteration gehört.

Beim *Ant Colony System*, bei dem nur die *Elitist Ant* Pheromon ablagert, macht es dagegen keinen Unterschied, ob die ungültigen Lösungen nicht beachtet oder bestraft werden (eine geeignet große Strafe vorausgesetzt), oder ob nur gültige Lösungen als *Elitist Ant* zugelassen werden: Letztendlich laufen alle diese Varianten darauf hinaus, dass das globale Pheromon-Update (Gleichung 6.9) durch die beste gültige Lösung, die bisher gefunden wurde, erfolgt, egal auf welchem Weg man dorthin gelangt.

6.3.5. Lokale Suche

Verschiedene Untersuchungen (z.B. Dorigo & Gambardella (1997) für das *Ant Colony System*) zeigen für das TSP und das Quadratische Zuordnungsproblem, dass die Kombination von Ameisenalgorithmen mit einer lokalen Suche die Qualität der Lösungen gehörig verbessert. Dies gilt umso mehr, wenn η ungünstig gewählt wurde (vgl. Cordón García *et al.* (2002) und Dorigo & Stützle (2004)). Da für das hier behandelte Problem η umgedeutet wurde (vgl. Abschnitte 6.3.2) ist zu erwarten, dass eine lokale Suche hier einen positiven Effekt auf die Lösungsfindung haben sollte.

Allerdings bringt diese lokale Suche auch einen höheren Rechenaufwand mit sich, da sie zusätzliche Auswertungen der Zielfunktion erfordert. Diese Auswertungen sind aber der zeitaufwendigste Teil der Optimierung. Es ist also eine Abwägung zu treffen, ob der Gewinn in der Lösungsgüte die Kosten in der Rechenzeit überwiegt.

In der Struktur des Algorithmus schließt sich die lokale Suche direkt an die Lösungskonstruktion (und gegebenenfalls -reparatur) an: Nachdem eine Ameise die Konstruktion einer Lösung abgeschlossen hat, wird in der Nachbarschaft dieser Lösung nach einer besseren (gültigen) Lösung gesucht[20]. Sollte eine solche gefunden werden, so wird die alte Lösung durch diese ersetzt. Das Pheromon-Update erfolgt dann entsprechend der neuen Lösung (vgl. Abbildung 6.9).

Zur Anwendung der Lokalen Suche muss zunächst die Nachbarschaft einer Lösung definiert werden. Für kombinatorische Probleme ist dies keinesfalls trivial. Für das Instandsetzungsproblem bietet sich folgende Nachbarschaftsdefinition an: Zwei Lösungen sind benachbart, wenn man, um von einer zur anderen zu gelangen entweder

- die Jahre der Instandsetzung zwischen zwei im Zeitplan berücksichtigten Bauwerken vertauschen muss, oder

[20] z.B. mit einem Bergsteiger-Algorithmus (vgl. Abschnitt 2.4.2)

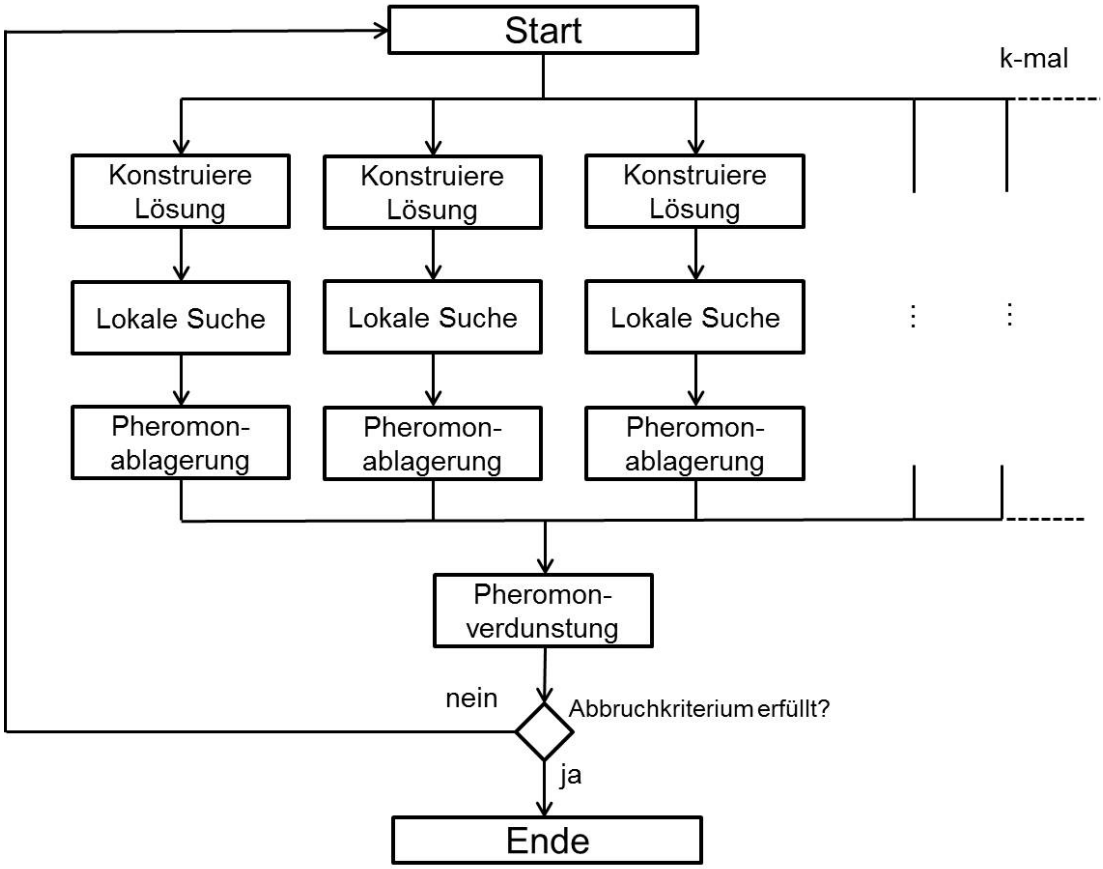

Abbildung 6.9.: Die lokale Suche im Ablauf des Ant Systems

- ein im Zeitplan berücksichtigtes Bauwerk durch ein nicht berücksichtigtes Bauwerk ersetzen muss.

Andere Definitionen sind durchaus denkbar.

Die Schwierigkeit, die sich nun ergibt, ist die Frage, welcher Anteil der Nachbarschaft abgesucht werden soll. Die Untersuchung der gesamten Umgebung einer Lösung verbietet sich durch ihre gewaltige Größe von selbst: Bei einer Problemgröße von 100 Bauwerken und einem Betrachtungszeitraum von 5 Jahren ergibt sich bei 10 Bauwerken pro Jahr ein Nachbarschaftsgröße von 3500[21]. Selbst wenn die ungültigen Lösungen von Haus aus ausgeschlossen werden, so ist die Anzahl immer noch gewaltig. Zum Vergleich: Bei einem Durchlauf des Algorithmus ohne lokale Suche mit $m = 50$ und 50 Generationen sind insgesamt 2500 Auswertungen der Zielfunktion nötig.

Folglich kann nur ein Teil der Nachbarschaft nach einer besseren Lösung abgesucht werden. Um auch hier die Bevorzugung bestimmter Teile des Suchraumes einzuschränken, bietet sich dabei eine zufallsbasierte Suche an: Ein zufälliges Bauwerk im Terminplan wird gewählt und mit diesem versucht, einen zulässiger Tausch mit einem, in einem anderen Jahr terminierten oder einem nicht berücksichtigten Bauwerk durchzuführen (siehe Abbildung 6.10). Jede mögliche durch so einen gültigen Tausch entstandene Lösung wird ausgewertet. Tritt dabei eine Verbesserung der Zielfunktion ein, so wird die alte Lösung durch die neue ersetzt.

Allerdings stellt sich auch hier die Frage nach dem Verhältnis zwischen Aufwand und Nutzen. Bei dem Beispiel aus Abbildung 6.10 hält sich der Aufwand noch in Grenzen, da nur sechs Nachbarlösungen untersucht werden müssen. Wäre aber bei der zufälligen Wahl nicht Bauwerk 9, sondern Bauwerk 74 ausgewählt worden, um von dort aus die Nachbarschaft abzusuchen, kämen zusätzlich zu den sechs noch weitere 20 Nachbarlösungen aus den Jahren 4 und 5 sowie 50 (bei einer Gesamtzahl von 100 Bauwerken) aus den nicht berücksichtigten Brücken dazu, in der Summe müssten also 76 Nachbarlösungen untersucht werden. Dies führt zu einem stark erhöhten Zeitaufwand: Rechnet man 2s für einen Aufruf des Verkehrssimulators, also 10s für die Auswertung eines gesamten Terminplanes mit fünf Jahren, so benötigt man zur Untersuchung der Nachbarschaft allein dieser Lösung 760s = 12min 40s. Im Schnitt lassen sich mit einer Brücke aus dem Beispiel in Abbildung 6.10 27 gültige Tauschoperationen durchführen. Ein möglicher Kompromiss, um nicht ganz auf die Lokale Suche zu verzichten, diese aber auch nicht zur Hauptsuchfunktion des Algorithmus zu machen, ist es, nur einige der möglichen Tauschoperationen, zufällig ausgewählt, zu untersuchen.

[21] Jedes der 10 im ersten Jahr terminierten Bauwerke kann mit den 40 restlichen Bauwerken des Zeitplans sowie mit den 50 nicht berücksichtigten Bauwerken vertauscht werden ($= 90 * 10 = 900$ Möglichkeiten). Für die 10 Bauwerke im 2. Jahr ergibt nur noch der Tausch mit den 30 Bauwerken der Jahre 3 bis 5 sowie den 50 nicht berücksichtigeten Bauwerken eine neue Lösung ($= 80 * 10 = 800$ Möglichkeiten). Entsprechendes gilt für die jeweils 10 Bauwerke der Jahre 3 bis 5. Zusammengefasst: $900 + 800 + 700 + 600 + 500 = 3500$ Möglichkeiten.

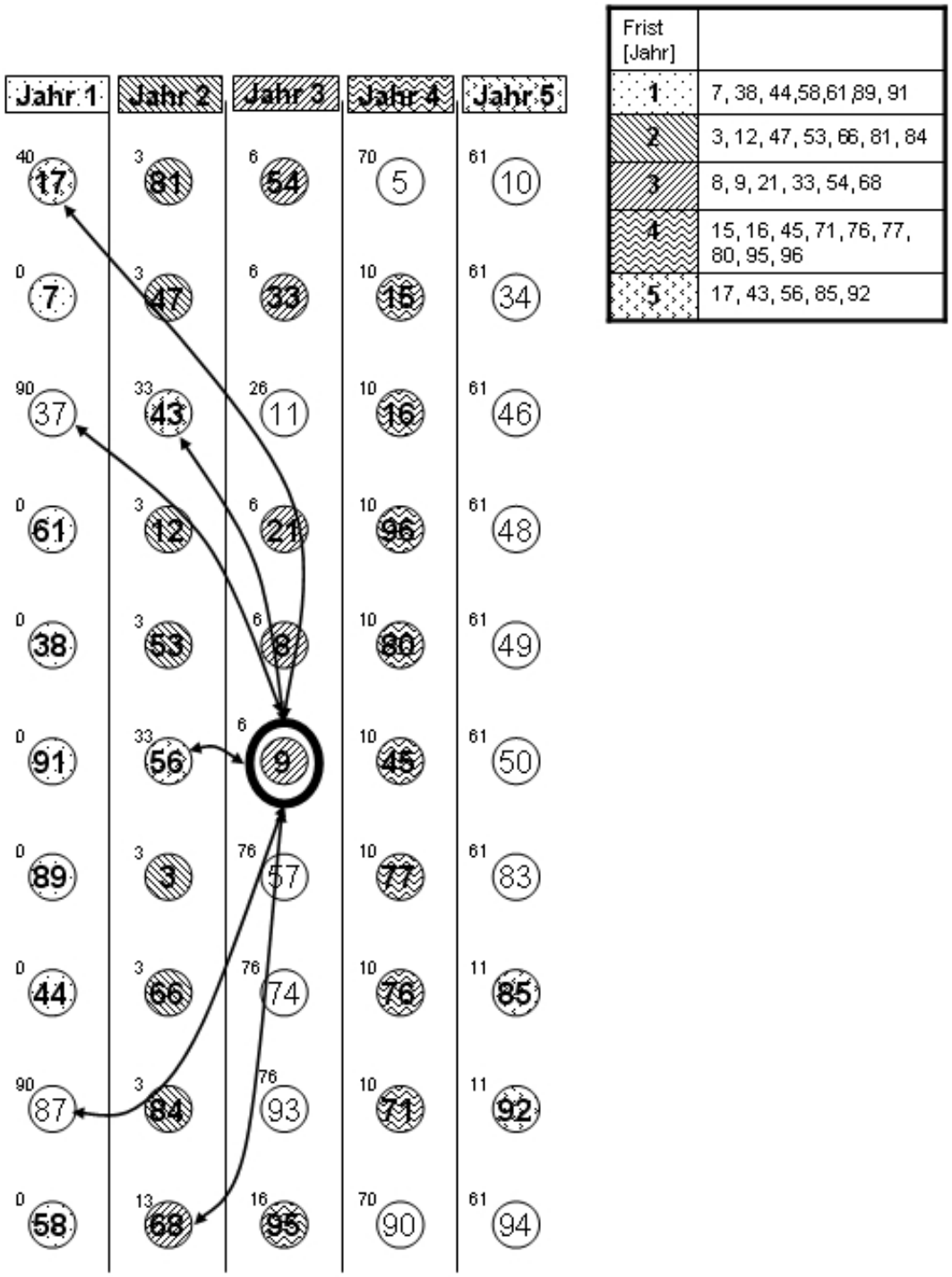

Abbildung 6.10.: Lokale Suche: Bauwerk 9 wurde zufällig ausgewählt. Es kann mit sechs anderen Bauwerken so vertauscht werden, dass eine neue gültige Lösung entsteht. Die kleinen Zahlen links oberhalb der Bauwerksnummern geben an, wie viele gültige Tauschoperationen mit diesen möglich wären.

6.3.6. Zusammenfassung

Um Ameisenalgorithmen auf das Instandsetzungsproblem anwenden zu können sind einige Anpassungen erforderlich. Mit die gravierendste Einschränkung der Ameisenalgorithmen ist dabei, dass in ihnen das Auftreten ungültiger Lösungen nicht vorgesehen ist. Bisherige Umsetzungen dieser Algorithmen gehen davon aus, dass ungültige Lösungen schon während der Lösungskonstruktion vermieden werden können.

Aufgrund der Struktur des Instandsetzungsproblems ist dies hier jedoch nicht möglich. Im Rahmen dieser Arbeit wurden verschiedene Maßnahmen zur Behandlung von ungültigen Lösungen in Ameisenalgorithmen entwickelt. Neben Methoden, die von den Genetischen Algorithmen übernommen wurden (vor allem Strafen und Reparaturfunktionen), sind dies vor allem die Umdeutung des mangels einer geeigneten Heuristik frei gewordenen Parameters η sowie der *Elitist Ant* zur Steuerung in gültige Bereiche.

Eine ausführliche Bewertung dieser Maßnahmen sowie der untersuchten Algorithmen findet sich in Abschnitt 7.2.2.

6.4. Ameisenalgorithmen für mehrkriterielle Probleme

Für Ameisenalgorithmen zur Anwendung auf multikriterielle Probleme sprechen ähnliche Argumente wie für Genetische Algorithmen (vgl. Abschnitt 5.4): Wie diese arbeiten sie auf einer Menge von Lösungen, was es ermöglicht, verschiedene Bereiche der Pareto-Front gleichzeitig abzusuchen.

Als Vorteil von Genetischen Algorithmen bei der Suche nach der Pareto-Front wurde in Abschnitt 5.4 zudem die Kreuzung genannt, die es ermöglicht die für jeweils ein Kriterium guten Eigenschaften zweier verschiedener Lösungs-Individuen so in ein Nachwuchsindividuum zu überführen, dass dieses für beide Kriterien gut angepasst ist. Bei Ameisenalgorithmen gibt es zwar keine wirkliche Kreuzung zweier Lösungen, durch die Konstruktion der Lösungen als Pfad, der einer Pheromonspur folgt, sind die neuen Lösungen einer Iteration dennoch implizit Kombinationen der Bestandteile guter Lösungen der vorangegangenen Iterationen. Daher ist es auch hier möglich, dass bei der Konstruktion neuer Lösungen Bestandteile, die sich positiv auf Kriterium 1 auswirken mit solchen kombiniert werden, die sich positiv auf Kriterium 2 auswirken.

Obwohl Ameisenalgorithmen an sich noch ein sehr neuer Ansatz sind, wurden in den letzten Jahren eine Vielzahl von Algorithmen entwickelt, die diese auf mehrkriterielle Probleme anwenden. Wie auch für die Genetischen Algorithmen werden an dieser Stelle nur die in dieser Arbeit verwendeten Algorithmen näher

ausgeführt. Für einen umfassenden Überblick siehe Garcia-Martinez *et al.* (2007), Angus & Woodward (2009) und Leguizamón & Coello Coello (2011).

Die verwendeten Algorithmen wurden daher augewählt, da beide darauf abzielen, die gesamte Paretofront bei der Suche abzudecken. Andere Ansätze dagegen, z.B. die erste Version von MACS (Gambardella *et al.*, 1999), COMPETANTS (Doerner *et al.*, 2001a,b) und MOACOM (Gravel *et al.*, 2002), setzen voraus, dass die einzelnen Zielkriterien von vornherein bekannte Prioritäten besitzen, so dass gezielt Lösungen gesucht werden, die für das bevorzugte Kriterium gute Werte liefern. Ein ähnliches Problem ergibt sich für MOAQ (Mariano & Morales, 1999): Hier werden die Prioritäten zwar nicht von außen vorgegeben, durch die Abarbeitung der einzelnen Zielkriterien in einer (zufällig bestimmten) festen Reihenfolge kommt es dennoch zu einer Bevorzugung bestimmter Bereiche der Paretofront (García-Martínez *et al.*, 2004).

Auch gegen die Verwendung einiger anderer Ansätze, die auf die Suche über die ganze Paretofront abzielen, auf das Instandsetzungsproblem sprechen verschiedene Argumente aus dessen Problemstruktur: So setzt die Erweiterung von MACS (Barán & Schaerer, 2003) voraus, dass für jedes Zielkriterium eine eigene Heuristik vorhanden ist. Diese gehen dann in verschiedene η-Matrizen ein, während nur eine einzige Pheromonmatrix verwendet wird. Da im hier behandelten Instandsetzungsproblem für keines der Kriterien eine geeignete Heuristik zur Bestimmung von η bekannt ist, scheidet dieser Ansatz aus.

Gegen die Verwendung des populationsbasierten Ansatzes von Guntsch & Middendorf (2003) spricht, dass dieser auf stark beschränkten Problemen, wie dem Instandsetzungsproblem, zu einer zu geringen Diversität führt, und damit nur einzelne Bereiche der Pareto-Front absucht.

6.4.1. Bi-Criterion Ant

Das Bi-Criterion Ant System (Iredi *et al.*, 2001) wurde, wie sein Name sagt, ursprünglich für zwei-kriterielle Probleme entwickelt. Seine Idee basiert wie der MOGA-Jas (Abschnitt 5.4.2) auf der Beschreibung der Paretofront als die Menge der Optima aller gewichteten Fitnessfunktionen.

Anders als bei dem erwähnten Genetischen Algorithmus werden die Gewichte hier nicht jede Iteration zufällig geändert. Stattdessen bekommt jede Ameise k der Kolonie über einen speziellen Wichtungsfaktor λ_k ihren eigenen Sektor des Lösungsraumes zugewiesen. Für jedes Zielkriterium wird eine eigene Pheromonmatrix erstellt. Für das ursprüngliche Bi-Criterion Ant System für ein zwei-kriterielles Optimierungsproblem berechnet sich dann die Wahrscheinlichkeit der Wahl von Kante ij (mit Pheromonablagerung auf den Kanten)

$$p_{ij}^k = \frac{[\tau_{ij}]^{\lambda\alpha} \cdot [\tau'_{ij}]^{(1-\lambda)\alpha} \cdot [\eta_{ij}]^{\lambda\beta} \cdot [\eta'_{ij}]^{\lambda\beta}}{\sum_{l \in N_i^k} [\tau_{il}]^{\lambda\alpha} \cdot [\tau'_{il}]^{(1-\lambda)\alpha} \cdot [\eta_{il}]^{\lambda\beta} \cdot [\eta'_{il}]^{\lambda\beta}} \quad \text{, für } j \in N_i^k \tag{6.18}$$

wobei τ das Pheromon für das erste und τ' das Pheromon für das zweite Zielkriterium beschreibt. λ_k berechnet sich für Ameise k als

$$\lambda_k = \frac{k-1}{m-1} \tag{6.19}$$

Eine Pheromon-Ablagerung wird durch alle nicht-dominierten Lösungen einer Iteration vorgenommen (vorher findet, wie im einkriteriellen Fall, eine Verdunstung auf allen Kanten entsprechend Gleichung 6.2 statt). Dabei wird auf jeder Pheromonmatrix in jeder Iteration die gleiche Menge Pheromon abgelagert. $\Delta\tau_{ij}$ für eine zu einer nicht-dominierten Lösung gehörenden Kante ij berechnet sich dabei als

$$\Delta\tau_{ij} = \frac{1}{v} \tag{6.20}$$

mit v der Anzahl aller ablagernden Ameisen.

Dieses $\Delta\tau_{ij}$ wird zum entsprechenden Eintrag der Pheromonmatrix hinzuaddiert, für deren Zielkriterium die jeweilige Lösung „gut“ ist. Für welches Kriterium dies der Fall ist, bestimmt sich folgendermaßen: Nach der Lösungskonstruktion jeder Iteration wird für jedes Kriterium der höchste und der niedrigste erzielte Wert bestimmt. Dieses Intervall wird dann in zwei Teile geteilt. Die Lösungen, die im besseren dieser Teile liegen (dem mit den niedrigeren Werten für ein Minimierungsproblem, der mit den größeren Werten für ein Maximierungsproblem), sind dann diejenigen, die auf der entsprechenden Pheromonmatrix Pheromon ablagern. Lösungen, die im zentralen Bereich der Pareto-Front liegen, lagern also auf beiden Matrizen ab, während Lösungen aus den Randbereichen dies nur auf einer tun.

6.4.1.1. Anpassungen für das multikriterielle Instandsetzungsproblem

Zur Anwendung des Bi-Criterion Ant System auf das Instandsetzungsproblem sind einige Anpassungen notwendig. Zum einen handelt es sich bei dem Instandsetzungsproblem nicht um ein zwei-kriterielles, sondern um ein drei- oder allgemein n-kriterielles Problem. Zum anderen ist für keines dieser Kriterien ein heuristisches Entscheidungskriterium, das die Attraktivität der Wahl von Bauwerk j im Jahr i bewertet, bestimmbar, da alle Kriterien von der Kombination der einzelnen Bauwerke abhängen.

Letzteres lässt sich umgehen, indem man nur ein einziges η verwendet, das so, wie in Kapitel 6.3.2 beschrieben, die Suche in den gültigen Bereich steuert. Dieses wird dann natürlich nicht mit einem Faktor λ gewichtet.

Zur Übertragung des Konzepts auf ein n-kriterielles Problem wird das einzelne λ aus Gleichung 6.19 durch n λs $\lambda_1 \dots \lambda_n$ ersetzt[22]. Die einzelnen λs werden so

[22] Eigentlich handelt es sich auch für das zwei-kriterielle Problem um zwei λs: Nämlich λ_1 berechnet nach Gleichung 6.19 und $\lambda_2 = 1 - \lambda_1$

gewählt, dass sie sich für jede Ameise k zu 1 ergänzen. Für das hier behandelte drei-kriterielle Problem bestimmen sie sich als[23]

$$\lambda_1 = \frac{2k^2 - 3km + 2k + m^2 - m}{m^2 - m};\ \lambda_2 = \frac{2k - 2km + 2k^2}{m - \frac{m^2}{2}};\ \lambda_3 = \frac{2k^2 - km}{m^2 - 3m + 2} \tag{6.21}$$

Die Wahrscheinlichkeit der Wahl von Knoten j in der Spalte i bestimmt sich dann als

$$p_{ij}^k = \frac{\left[\tau_{ij}^1\right]^{\lambda_1\alpha} \cdot \left[\tau_{ij}^2\right]^{\lambda_2\alpha} \cdot \left[\tau_{ij}^3\right]^{\lambda_3\alpha} \cdot \left[\eta_{ij}\right]^{\beta}}{\sum_{l \in N_i^k} \left[\tau_{il}^1\right]^{\lambda_1\alpha} \cdot \left[\tau_{il}^2\right]^{\lambda_2\alpha} \cdot \left[\tau_{il}^3\right]^{\lambda_3\alpha} \cdot \left[\eta_{il}\right]^{\beta}} \qquad \text{, für } j \in N_i^k \tag{6.22}$$

Als letzte Ergänzung ist noch der Umgang mit ungültigen Lösungen zu erwähnen. Da für die meisten Probleme, auf die Ameisenalgorithen angewandt werden, ungültige Lösungen durch die Art der Lösungskonstruktion vermieden werden können, ist auch für das Bi-Criterion Ant System keine Behandlungsmöglichkeit für ungültige Lösungen vorgesehen. Da ungültige Lösungen nicht Teil des endgültigen Optimierungsergebnis sein dürfen, liegt es nahe, ungültige Lösungen immer als dominiert zu betrachten. Dies kann aber zu Schwierigkeiten führen, da nur die nicht-dominierten Lösungen einer Iteration überhaupt Pheromon ablagern. Besonders wenn in einer Iteration überhaupt keine gültige Lösung gefunden wird, bedeutet dies, dass diese Iteration keine neuen Informationen liefert. Eine mögliche Abhilfe für dieses Problem ist es, in solchen Iterationen, in denen keine gültigen Lösungen vorhanden sind, eine Pheromonablagerung für die nicht-dominierten ungültigen Lösungen dieser Iteration durchzuführen.

6.4.2. Pareto Ant Colony Optimization

Auch die Pareto Ant Colony Optimization (PACO, Doerner *et al.* (2004)) basiert auf gewichteten Fitnessfunktionen. Anders als das Bi-Criterion Ant System baut sie aber nicht auf dem klassischen Ant System (Kapitel 6.2.1) sondern auf dem Ant Colony System (Kapitel 6.2.4) auf. Mit diesem teilt sie den Vorteil, gute Lösungen mit einer geringen Anzahl an Zielfunktionsauswertungen zu finden, da sie mit sehr kleinen Kolonien der Größe $m = 10$ arbeitet.

Wie beim Bi-Criterion Ant System hat jede Ameise k bei der PACO ihre eigenen Wichtungsfaktoren w_k^ζ für die einzelnen Zielkriterien ζ. Anders als dort sind diese jedoch nicht gleichmäßig verteilt, sondern werden, ähnlich wie bei dem Genetischen Algorithmus von Ishibuchi & Murata (1998) (vgl. Absatz 5.4.2), jede Iteration neu zufällig bestimmt.

Zur eigentlichen Lösungskonstruktion arbeitet die PACO ebenfalls mit eigenen Pheromonmatrizen für die einzelnen Zielkriterien. Die Pheromonmengen τ^ζ auf diesen verschiedenen Matrizen werden dann mit den entsprechenden w_k^ζ für jede

[23] Die Ameisen haben Nummern zwischen $0 \leq k < m$.

Ameise k gewichtet. Wie beim ACS wird mit einer Wahrscheinlichkeit q_0 der Knoten j entsprechend

$$j = \arg\max \left\{ \left[\sum_{\zeta} (w_k^{\zeta} \tau_{ij}^{\zeta}) \right]^{\alpha} \cdot [\eta_{ij}]^{\beta} \right\} \tag{6.23}$$

gewählt. Ansonsten erfolgt die Wahl probabilistisch mit den Wahrscheinlichkeiten nach

$$p_j = \frac{\left[\sum_{\zeta} (w_k^{\zeta} \tau_{ij}^{\zeta}) \right]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k} \left[\sum_{\zeta} (w_k^{\zeta} \tau_{il}^{\zeta}) \right]^{\alpha} \cdot [\eta_{il}]^{\beta}} \tag{6.24}$$

Da im mehrkriteriellen Fall eine größere Diversität der Lösungen nötig ist, als im einkriteriellen Fall, muss q_0 gegenüber dem ACS gesenkt werden. Doerner *et al.* (2004) empfehlen $q_0 = 0.4$.

Ebenfalls wie beim ACS findet keine globale Pheromonverdunstung statt, sondern die Ameisen entfernen beim Laufen Pheromon entsprechend Gleichung 6.10 auf allen Pheromonmatrizen. Pheromonablagerung erfolgt für die jeweils beiden besten Lösungen für jedes Zielkriterium ζ entsprechend Gleichung 6.9:

$$\tau_{ij}^{\zeta} \leftarrow (1 - \rho)\tau_{ij}^{\zeta} + \rho \Delta \tau_{ij}^{\zeta}. \tag{6.25}$$

$\Delta \tau_{ij}^{\zeta}$ bekommt dabei den Wert 10, wenn ij in der besten Lösung für Kriterium ζ vorhanden ist, 5, wenn es in der zweitbesten Lösung, und 15, wenn es in der besten und der zweitbesten Lösung vorhanden ist. Ist ij in keiner der beiden Lösungen vorhanden, so hat $\Delta \tau_{ij}^{\zeta}$ den Wert 0.

6.4.2.1. Anpassungen für das mehrkriterielle Instandsetzungsproblem

Aufgrund des stark zerfaserten gültigen Lösungsraumes beim Instandsetzungsproblem, ist auch für die PACO eine Ergänzung notwendig, die dem Algorithmus erlaubt, mit ungültigen Lösungen zu arbeiten.

Dabei muss unterschieden werden, wann eine solche ungültige Lösung auftritt: Sind unter den innerhalb einer Iteration gefundenen Lösungen auch gültige vorhanden, so können die ungültigen Lösungen bei der Bestimmung der besten Lösungen für jedes Zielkriterium ignoriert werden. Die Ameisen entfernen zwar bei ihrer Konstruktion Pheromon entsprechend Gleichung 6.10, können aber nicht zu einer ablagernden Ameise werden (vergleiche die Behandlung der Elitist Ant in Kapitel 6.3.4). Andererseits kann es auch vorkommen, dass in einer Iteration gar keine gültigen Lösungen gefunden wurden. Damit die Information aus diesen Iterationen nicht vollständig verloren geht, bietet es sich an, in diesem Fall den jeweils zwei besten ungültigen Ameisen für jedes Zielkriterium zu erlauben, Pheromon abzulagern.

6.4.3. Zusammenfassung

Für die Behandlung des mehrkriteriellen Instandsetzungsproblems mit Ameisenalgorithmen wurden in dieser Arbeit zwei verschiedene Ansätze ausgewählt: Das Bi-Criterion Ant System (Iredi *et al.*, 2001) und die Pareto Ant Colony Optimization (Doerner *et al.*, 2004).

Beide Methoden müssen für die Anwendung auf das Instandsetzungsproblem angepasst werden. So ist das Bi-Criterion Ant System ursprünglich nur auf zweikriterielle Probleme ausgelegt und muss mit Ergänzugen für eine Behandlung von mehr Kriterien ergänzt werden. Für beide untersuchte Methoden ist ein Auftreten ungültiger Lösungen nicht vorgesehen, so dass auch hier Anpassungen notwendig sind.

7. Ergebnisse

7.1. Testszenarien

Zur Untersuchung der Qualität der in den Abschnitten 5.3 und 6.3 (für das einkriterielle Problem) sowie 5.4 und 6.4 (für das mehrkriterielle Problem) beschriebenen Algorithmen, wurden mehrere Testszenarien erstellt. Diese werden in diesem Abschnitt kurz vorgestellt, eine ausführliche Beschreibung der einzelnen Szenarien findet sich im Anhang.

Das Szenario 1 (Abbildung 7.1 und Anhang A.1) wurde erstellt, um zu testen, wie nahe dem globale Optimum die einzelnen Algorithmen kommen. Da es aufgrund der komplexen Problemstruktur und Zielfunktion nicht möglich ist, ein Szenario zu konstruieren, dessen Optimum bekannt ist, wurde das Szenario so klein gewählt, dass sein Optimum in annehmbarer Zeit durch Vollständige Enummeration zu ermitteln ist.

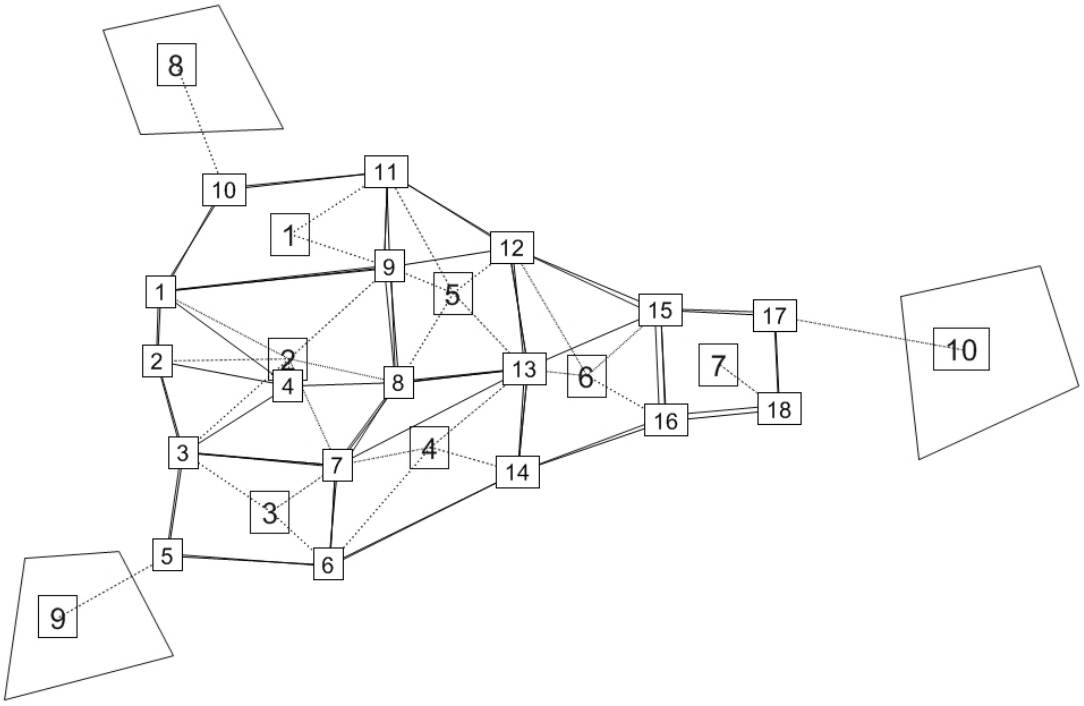

Abbildung 7.1.: Netz von Szenario 1

Gewählt wurde eine Problemgröße von $n = 30$, $y = 5$ und $l = 4$. Nach Gleichung 3.1 ergibt sich daraus eine Problemgröße von $|X| = 9.18 * 10^{18}$. Für die Szenarioerstellung mit bekanntem Optimum ist jedoch keine Auswertung aller dieser Szenarien nötig: hierzu genügt es, alle einzelnen Viererkombinationen (also die einzelnen Jahre des Terminplans) auszuwerten und erst nachträglich einen vollständigen Terminplan daraus so zu erstellen, dass dieser derjenige mit dem geringsten Verkehrseinfluss ist, in dem jedes Element nur einmal vorkommt. Erst im Nachhinein werden dann die einzelnen Bauwerke mit Instandsetzungsfristen und -kosten so versehen, so dass dieser Terminplan gültig ist. Damit müssen nur 27405 verschiedene Szenarien ausgewertet werden, was, bei Annahme von 5s pro Auswertevorgang, innerhalb von nur 38h möglich ist.

Szenario 2 (Abbildung 7.2 und Anhang A.2) wurde erstellt, um die Ergebnisse leichter nachvollziehbar zu machen. Zudem konnte auf diesem Netz leicht überprüft werden, ob sich der Verkehrssimulator entsprechend den Anforderungen verhält. Zu diesem Zweck wurde ein hochgradig symmetrisches Netz erstellt, in dem alle Straßen die selbe Kapazität haben. Diese wurde so gewählt, dass das Netz auch im ungestörten Zustand leicht überlastet ist, um die Auswirkungen von Störungen zu verstärken. Quell- und Zielbezirke wurden so angeordnet, dass sich im ungestörten Netz eine ebenfalls symmetrische Belastung einstellt[1].

Jeder Straße im Netz wurde ein Bauwerk zugeordnet. Damit ergibt sich für das Szenario 2 $n = 112$. Die übrigen Größen wurden gewählt als $y = 5$ und $l = 10$. Die Größe des Lösungsraumes ergibt sich damit nach Gleichung 3.1 zu $9.97 * 10^{63}$. Um zu gewährleisten, dass in jedem Jahr des Terminplans, auch dem ersten, eine größere Anzahl an instandzusetzenden Bauwerken frei wählbar ist, wurden die Instandsetzungsfristen so gesetzt, dass in jedes der ersten fünf Jahre jeweils die Frist für sechs Bauwerke fällt.

Auf dem Netz wurden die kritischen Bauwerke achssymmetrisch verteilt, um die Ergebnisse der Optimierung leichter nachvollziehbar zu machen. Zusätzlich wurden sie so angeordnet, dass die Bauwerke mit Instandsetzungsfrist 1 dort liegen, wo die stärkste Belastung auftritt. Es ist zu erwarten, dass der verkehrsgünstigste Terminplan für das erste Jahr neben den kritischen Bauwerken mit Frist = 1 auch einen Großteil der weiteren stark belasteten kritischen Bauwerke (also die Bauwerke in den beiden äußersten Straßenzügen) enthält. Basierend auf dieser

[1] Je nach verwendetem Umlegungsalgorithmus ist diese Symmetrie mehr oder weniger stark ausgeprägt. Die Symmetrie zeigt sich aber bei allen getesteten Umlegungsverfahren.

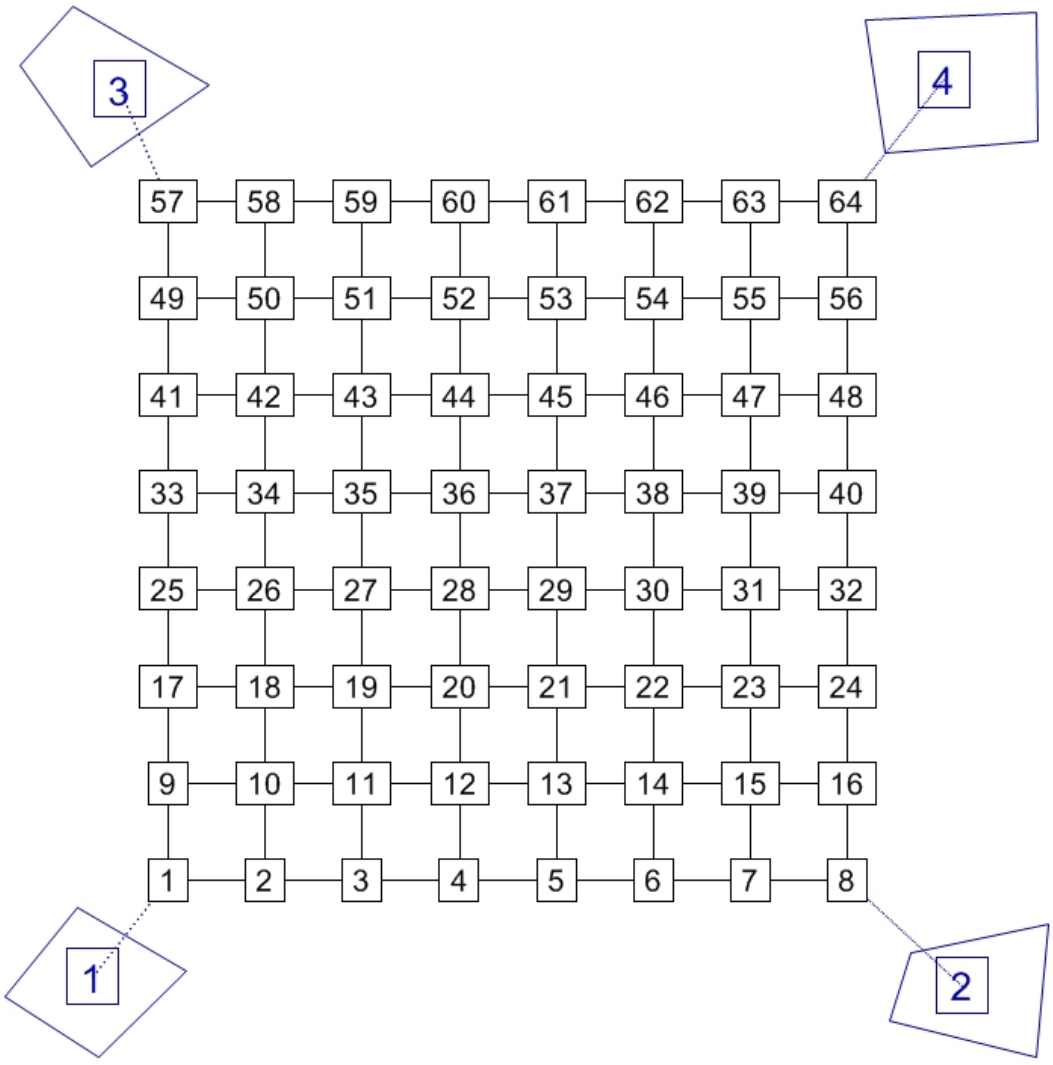

Abbildung 7.2.: Netz von Szenario 2

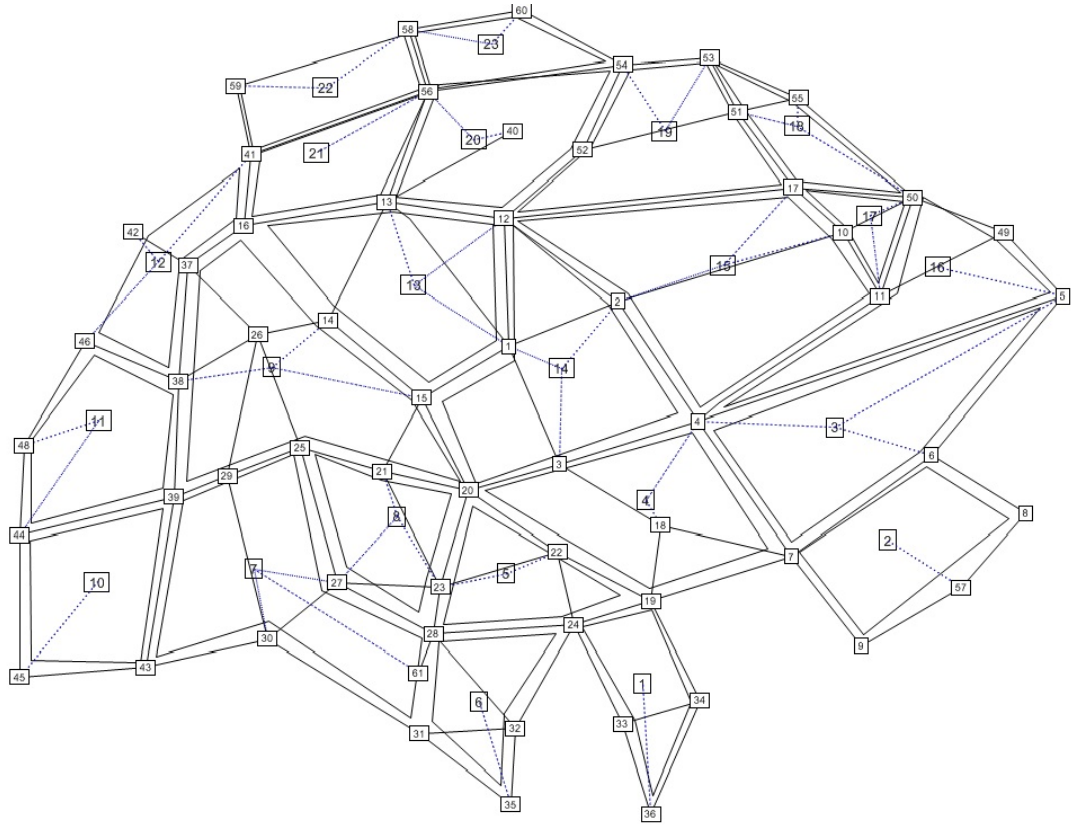

Abbildung 7.3.: Netz von Szenario 3

Überlegung wurde auch eine händische Lösung für das Problem erstellt[2]. Die Instandsetzungskosten der einzelnen Bauwerke wurden zufällig zugewiesen.

Die Szenarien 3 und 4 wurden entwickelt, um die Anwendbarkeit der entwickelten Algorithmen auf realistische Szenarien zu zeigen. Szenario 3 (Abbildung 7.3 und Anhang A.3) stellt dabei ein fiktives Netz, ähnlich dem aus Szenario 1, dar, allerdings enthält es $n = 100$ Bauwerke. Wie im Szenario 2 gilt auch hier $y = 5$ und $l = 10$. Instandsetzungsfristen und -kosten wurden zufällig gesetzt, wobei nur darauf geachtet wurde, dass die Anzahl der kritischen Bauwerke niedrig genug ist, um gültige Lösungen zuzulassen.

Szenario 4 (Abbildung 7.4 und Anhang A.4) ist ein relitätsnahes Szenario. Das Netz ist ein Ausschnitt des Münchner Innenstadtnetzes, Kapazitäten und Belastungen wurden möglichst nah an reale Daten angepasst. Einzig die Instandset-

[2]Beginnend mit dem Jahr 1 wurden zunächst die kritischen Bauwerke für dieses Jahr festgelegt. Die noch freien Elemente dieses Jahres wurden dann mit Bauwerken aufgefüllt, die einerseits die Kostenrandbedingung nicht verletzten andererseits aus der Betrachtung der Simulationsergebnisse am ungestörten Netz als geeignet erschienen, wobei Bauwerke mit niedriger Instandsetzungsfrist bevorzugt gewählt wurden. So wurde für alle weiteren Jahre verfahren, bis alle kritischen Bauwerke gesetzt waren. Die restlichen Jahre wurden mit im ungestörten Netz un- oder nur sehr schwach belasteten Bauwerken gefüllt. Insgesamt fanden mehrere Versuche zur händischen Erstellung eines optimalen Terminplanes statt.

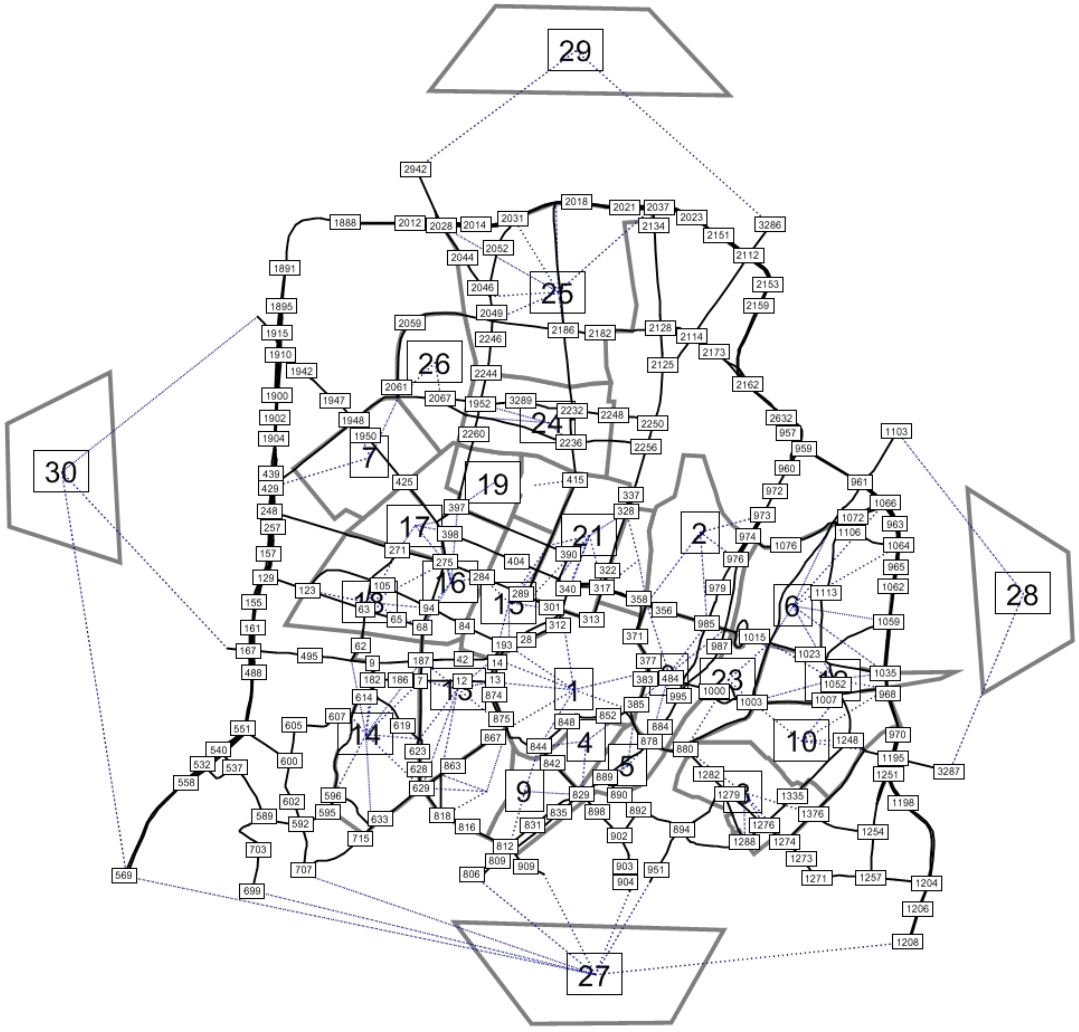

Abbildung 7.4.: Netz von Szenario 4

zungsfristen und -kosten der betrachteten Bauwerke, ebenso wie die Daten für Instandsetzungsarbeiten von dritter Seite, wurden frei erfunden.

Im Folgenden werden als Ergebnisse der Optimierungsläufe für das Zielkriterium „Minimaler Einfluss auf den Verkehr“ die Absolutwerte der Verkehrsbelastung in Fahrzeug-Sekunden [Fzgs] angegeben. Versuche, diese Ergebnisse gegenüber den Werten für das ungestörte Netz zu normieren, erwiesen sich als wenig aussagekräftig, da auch die Stärke der Abweichung zwischen gestörtem und ungestörtem Netz stark vom Szenario abhängt. So vergrößert sich die Verkehrsbelastung im Szenario 1 im gestörten Netz im Optimum nur um etwa 4-5%, während sie in Szenario 2 fast verdoppelt wird.

	Matrix	**Permutation**
Mittelwert [Fzgs*10^8]	1.9143	1.9166
Beste Lösung [Fzgs*10^8]	1.9119	1.9112
Standardabweichung [%]	0.09	0.30

Tabelle 7.1.: Ergebnisse aus jeweils fünf Testläufen über 100 Generationen auf Szenario 1. Das globale Optimum liegt bei $1.9089 * 10^8$ Fzgs (Fahrzeugsekunden).

7.2. Einkriterielles Problem

7.2.1. Genetische Algorithmen

7.2.1.1. Einfluss der gewählten Repräsentation

In Abschnitt 5.3.1 wurden verschiedene Repräsentationen für das Problem der Optimierung von Instandsetzungszeitplänen vorgestellt. Die beiden in Abschnitt 5.3.1.1 beschriebenen Repräsentationen wurden verworfen, da sie zu viele ungültige Lösungen erzeugen. Sie bieten keine Kontrolle darüber, wie häufig die einzelnen Bauwerke im Betrachtungszeitraum (in der Regel sollte dies einmal geschehen) und wie viele Bauwerke pro Jahr instand gesetzt werden.

Untersucht wurden daher nur die beiden in Abschnitt 5.3.1.2 beschriebenen Repräsentationen, Matrix- und Permutations-Repräsentation. Für beide Repräsentationen wurden jeweils 5 Testläufe auf dem Testszenario 1 über 100 Generationen durchgeführt. Die Populationsgröße betrug jeweils 100, ebenso die Größe der Elternmenge. Die Selektion sowohl der Eltern als auch die der Überlebenden erfolgte über Turniere der Größe 2. Für die Matrix-Repräsentation wurde eine Ein-Punkt-Kreuzung gewählt. Die Ergebnisse sind auch in Tabelle 7.1 dargestellt.

Beide Repräsentationen zeigten dabei ein sehr ähnliches Verhalten: Mit beiden wird innerhalb der 100 Generationen eine Lösung erreicht, die sehr nahe am bekannten Optimum liegt (im Mittel 0.3% über dem Optimum für die Matrix-Repräsentation bzw. 0.4% für die Permutations-Repräsentation). Die Matrix-Repräsentation erreicht dabei im Schnitt geringfügig bessere Ergebnisse. Dies zeigt, dass die befürchtete Bevorzugung bestimmter Lösungsraumgebiete durch diese Repräsentation nicht gegeben ist. Das geringfügig schlechtere Abschneiden der Permutations-Repräsentation lässt sich damit erklären, dass in ihr wesentlich mehr verschiedene Genotypen für den selben Phenotyp existieren als in der Matrix-Repräsentation[3]. Beide Repräsentationen zeigen sich dennoch gleichermaßen gut geeignet zur Lösung des Problems.

[3]Während in der Matrix-Repräsentation der selbe Phenotyp durch $(y*l!)$ Genotypen repräsentiert wird (unterschiedliche Reihenfolge der Bauwerke innerhalb eines Jahres) gibt es in der Permutationsrepräsentation pro Phenotyp $((y*l!)*(n-yl)!)$ Genotypen (Reihenfolge der nicht berücksichtigten Bauwerke).

	Matrix	**Matrix o. Reparatur**	**Permutation**
Mittelwert [Fzgs*10^8]	6.0635	6.2307	6.0705
Beste Lösung [Fzgs*10^8]	6.0526	6.0613	6.0597
Standardabweichung [%]	0.13	1.81	0.19

Tabelle 7.2.: Ergebnisse aus jeweils zehn Testläufen über 50 Generationen auf Szenario 2. Die beste bekannte Lösung liegt bei $6.0526 * 10^8$ Fzgs (Fahrzeugsekunden), eine händisch konstruierte Lösung erreichte $6.0811 * 10^8$ Fzgs.

Die Eignung zeigt sich auch bei Anwendung auf das Testszenario 2. Hier wurden jeweils 10 Testläufe pro Repräsentation über 50 Generationen durchgeführt. Populationsgröße und Größe der Elternmenge wurden jeweils auf 50 gesetzt. Wie auch in den oben beschriebenen Testläufen wurden auch hier zur Selektion der Eltern sowie der Überlebenden Turniere der Größe 2 durchgeführt. Auch hier erfolgte die Kreuzung bei der Matrix-Repräsentation durch Ein-Punkt-Kreuzung.

Die Ergebnisse dieser Testläufe (vgl. Tabelle 7.2) entsprechen im Wesentlichen denen der Läufe auf dem Szenario 1: Mit beiden Repräsentationen wurden gute Lösungen gefunden. Da die Lage des Optimums für das Szenario 2 nicht bekannt ist, können die Werte nur mit der besten gefundenen Lösung sowie einer händisch entsprechend der Überlegungen aus Abschnitt 7.1 konstruierten Lösung verglichen werden. Die beste bekannte Lösung des Problems wurde mit einem Lauf des Genetischen Algorithmus mit Matrix-Repräsentation gefunden. Diese ist besser als die händisch konstruierte Vergleichslösung. Alle übrigen in den Läufen mit Matrix-Repräsentation gefundenen Lösungen liegen zwischen diesen beiden Werten, waren also alle ebenfalls besser als die händische Lösung. Wie beim Szenario 1 sind die in den Läufen mit Permutations-Repräsentation gefundenen Lösungen im Schnitt geringfügig schlechter als die aus den Läufen mit Matrix-Repräsentation (und teilweise schlechter als die konstruierte Vergleichslösung), liegen jedoch im gleichen Wertebereich.

Auffällig an den Läufen auf dem Szenario 2 ist die schnelle Konvergenz mit beiden Repräsentationen[4]: Während bei den Läufen mit der Matrix-Repräsentation auf Szenario 1 dazu über 40 Generationen benötigt werden (vgl. Abbildung 7.5), erreichen die Läufe mit beiden Repräsentationen im Szenario 2 schon nach etwa 13 Generationen Werte in der Nähe des letztlichen Endergebnis (vgl. Abbildung 7.6). Die Problemstruktur von Szenario 2 scheint leichter zu behandeln als die in Szenario 1. Was genau die Ursachen dafür sind, kann derzeit nicht festgestellt werden.

Zusammengefasst lässt sich sagen, dass die Repräsentation der Lösungen nur einen sehr geringen Einfluss auf die Lösungsgüte hat. Dies setzt jedoch voraus, dass die Kreuzungs- und Mutationsoperatoren in Anpassung an das Problem gewählt wurden. Damit befasst sich der nächste Abschnitt.

[4]Mit Konvergenz ist hier die Annäherung an ein (lokales) Optimum gemeint

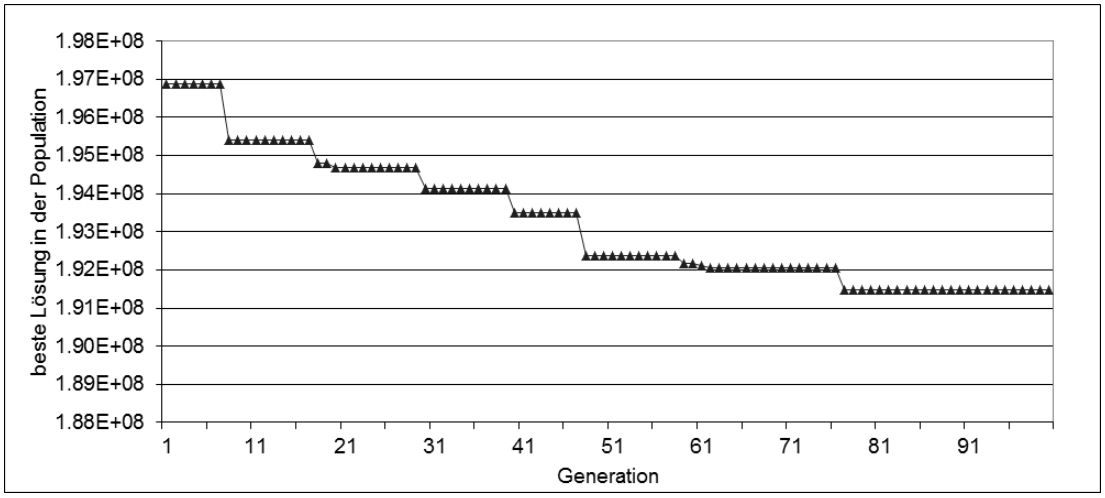

Abbildung 7.5.: Die jeweils beste Lösung der Population über die 100 Generationen eines Testlaufs eines Genetischen Algorithmus mit Matrix-Repräsentation auf Szenario 1. Die Populationsgröße, sowie die Größe der Elternmenge betragen jeweils 50, Selektion der Eltern und der Überlebenden erfolgte über Turnierselektion. Die Lösungsgüte nimmt über die Generationen nur sehr langsam zu.

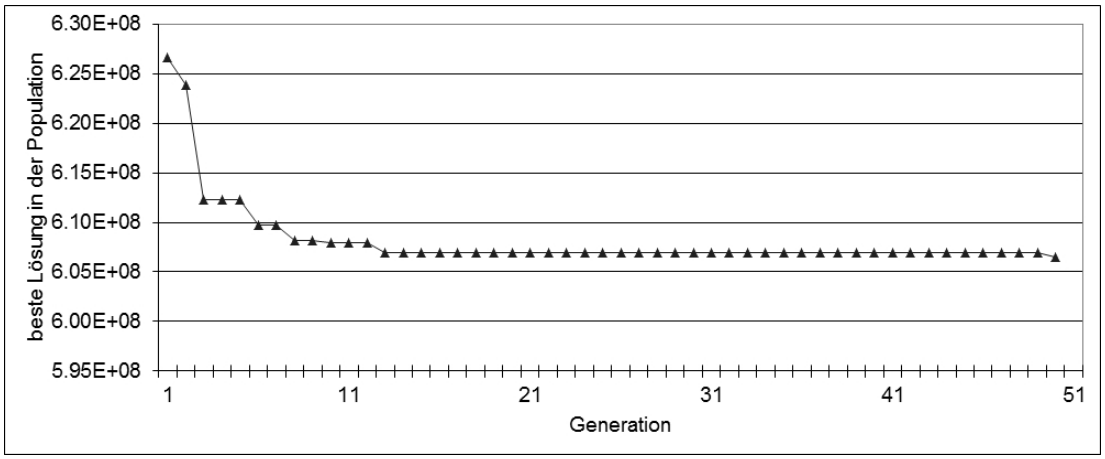

Abbildung 7.6.: Die jeweils beste Lösung der Population über die 50 Generationen eines Testlaufs eines Genetischen Algorithmus mit Matrix-Repräsentation auf Szenario 2. Die Populationsgröße, sowie die Größe der Elternmenge betragen jeweils 50, Selektion der Eltern und der Überlebenden erfolgte über Turnierselektion. Bereits nach wenigen Generationen werden sehr gute Lösungen gefunden.

7.2.1.2. Einfluss der genetischen Operatoren

Für die Matrix-Repräsentation wurden in Abschnitt 5.3.2.1 zwei verschiedene Kreuzungsoperatoren beschrieben: Die einfache Ein-Punkt-Kreuzung sowie ein Kreuzungsoperator, der ganze Jahre aus den jeweiligen Elternchromosomen übernimmt. Vom Gesichtspunkt des Respekts aus betrachtet ist der zweite Operator eigentlich vorzuziehen.

Wie jedoch schon in Abschnitt 5.3.2.1 erwähnt, zeigt dieser Operator in Kombination mit einem einfachen Mutationsoperator die Tendenz, bestimmte Regionen des Lösungsraumes zu bevorzugen und sich damit in lokalen Optima zu verfangen.

In Testläufen auf dem Szenario 3 mit einer Populations- und Elternmengengröße von 50 und Turnierselektion von Eltern und Überlebenden liefern alle 10 Läufe mit einer jahresweisen Kreuzung (und einfachem Mutationsoperator) eine von zwei verschiedenen Lösungen, die beide weit von den besten für dieses Szenario bekannten Lösungen entfernt sind. Sobald eine dieser Lösungen, häufig schon innerhalb der ersten wenigen Generationen, erreicht wurde, schafft es der Algorithmus nicht mehr, sich daraus zu befreien.

Auch in den Läufen mit Ein-Punkt-Kreuzung wurden diese lokalen Optima häufig gefunden, doch längst nicht alle Läufe blieben in ihnen hängen: Einige Läufe umgingen sie von Haus aus, bei anderen zeigte sich zumindest die Fähigkeit wieder von ihnen loszukommen. Gänzlich frei von derartigen Einflüssen zeigten sich jedoch nur die Läufe mit der Permutations-Repräsentation. Eine Möglichkeit, um dies auch für die Matrix-Repräsentation zu erreichen, wird im Abschnitt 7.2.1.4 beschrieben.

7.2.1.3. Einfluss verschiedener Selektionsverfahren

Zur Untersuchung der Selektionsverfahren wurden die verschiedenen Mechanismen zur Selektion der Überlebenden unter sonst gleichen Parametern (Populationsgröße 50, Größe der Elternmenge 50, Selektion der Eltern über Turnier, Matrixrepräsentation mit Ein-Punkt-Kreuzung) auf dem Szenario 1 getestet.

Bei einem altersbasierten Selektionsschema (vgl. gestrichelte Linie in Abbildung 7.7), bei dem die alte Generation komplett durch ihre Nachkommen ersetzt wird, zeigt sich dabei jeweils über den gesamten Lauf keine nennenswerte Verbesserung der gefundenen Ergebnisse. Einzelne bessere Lösungen gehen nach einigen Generationen wieder verloren, so dass die jeweils beste Lösung in der Population sich zwischendurch immer wieder über mehrere Generationen verschlechtert. Insgesamt bleibt die Suche durchgehend im gleichen Gebiet des Lösungsraumes hängen, da einzelne, durch Mutation entstandene, Ausreißer sofort wieder verloren gehen.

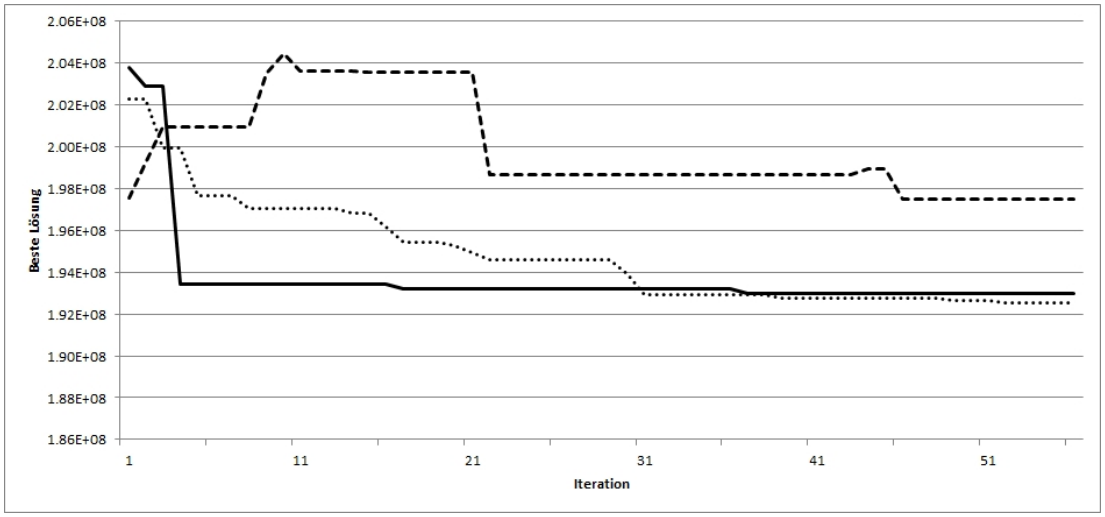

Abbildung 7.7.: Vergleich der unterschiedlichen Verfahren zur Selektion der Überlebenden auf Szenario 1. Die gestrichelte Linie zeigt einen Lauf mit altersbasierter Selektion. Hier gehen gute Lösungen schnell wieder verloren, eine Verbesserung der Ergebnisse findet nur selten statt, insgesamt bleibt die Qualität der Lösungen schlecht. Zwischen deterministischer Selektion nach Fitness (durchgezogene Linie) und Turnierselektion (gepunktete Linie) ist auf den ersten Blick kein Unterschied festzustellen. Mit der deterministischen Selektion findet nach schneller anfänglicher Verbesserung der Lösungen bald kaum noch eine weitere Qualitätsverbesserung statt, bis der Algorithmus in einem lokalen Optimum stagniert. Bei der Turnierselektion treten immer wieder kleinere Verbesserungen der Lösung auf.

Zwischen einer deterministischen Selektion (durchgezogene Linie in Abbildung 7.7), bei der die ψ besten Individuen aus der Gesamtmenge der alten Generation und der neu erzeugten Lösungen die nächste Generation bilden, und einer Turnierselektion (gepunktete Linie in Abbildung 7.7) zeigt sich in den ersten Generationen kein Unterschied: Mit beiden Selektionsschemata findet in diesen ersten Generationen eine rasche Verbesserung der jeweils besten Lösung der Population statt. Bei der deterministischen Selektion bleibt die Suche jedoch nach wenigen Generationen in einem lokalen Optimum hängen, dieses übernimmt schnell die gesamte Population. Bei Verwendung der Turnierselektion sind dagegen auch in späten Generationen immer noch Verbesserungen der Lösung möglich.

Der Grund dafür liegt darin, dass sich bei der deterministischen Selektion nach wenigen Generationen keine ungültigen Lösungen mehr in der Population befinden, da diese ja mit einer Strafe versehen einen schlechteren Fitnesswert haben als die meisten gültigen Lösungen. Die meisten Lösungen der Population befinden sich nun in dem gültigen Bereich, der die bisher beste gefundene Lösung umgibt. Da nun durch Kreuzungen der Individuen aus diesem Bereich immer mehr Kopien dieser besten Lösung und ihrer Nachbarschaft entstehen, sterben die außerhalb dieses Bereiches liegenden Lösungen bald aus, es sei denn, sie haben durch Zufall einen besseren Fitnesswert. Ist dies geschehen, ist es sehr unwahrscheinlich, dass später noch eine Lösung außerhalb des Bereichs des lokalen Optimums gefunden wird. Bei Verwendung der Turnierselektion ist dagegen immer ein gewisser Anteil an ungültigen und „schlechten“ Lösungen in der Population vorhanden, die es möglich machen, andere Regionen des Lösungsraumes zu untersuchen.

7.2.1.4. Behandlung ungültiger Lösungen

In den verschiedenen Testläufen zeigte sich auch die Bedeutung der Reparaturfunktionen (Abschnitt 5.3.3.4) sowie der vorgeschalteten Optimierung auf minimalen Verletzungsgrad der Randbedingungen (Abschnitt 5.3.3.3). Ganz ohne die Verwendung dieser beiden Mechanismen, rein mit Straffunktionen für ungültige Lösungen, brauchte der Algorithmus in allen Testläufen zwischen 30 und 40 Generationen, bis überhaupt eine gültige Lösung gefunden wurde.

Eine Auswertung mit dem Verkehrssimulator in diesen Generationen bedeutet einen erhöhten Zeitaufwand ohne weiteren Nutzen, da der treibende Faktor bei der Selektion der Elternmenge und der Überlebenden die Strafen sind. Da diese nach Gleichung 5.4 vom Grad der Randbedingungsverletzung abhängt, kann die eigentliche Zielfunktion wie in Abschnitt 5.3.3.3 beschrieben so lange vernachlässigt werden, bis die erste gültige Lösung gefunden wurde.

Lässt man nach diesem Vorlauf den Algorithmus so weiter laufen, dass die ungültigen Lösungen mit Strafen nach Gleichung 5.4 versehen werden, dann zeigt er Konvergenzverhalten und ist, nicht-deterministische Selektionsmechanismen vor-

ausgesetzt, in der Lage, das Optimum zu finden. Allerdings ist er dabei sehr langsam.

Beschleunigt werden kann die Suche nach dem Optimum durch die Verwendung der in Abschnitt 5.3.3.4 beschriebenen Reparaturfunktionen. Diese sorgen nicht nur dafür, dass in den meisten Fällen schon innerhalb der ersten Generation, spätestens aber in der zweiten, gültige Lösungen gefunden werden. Zusätzlich sorgen sie dafür, dass das Durchqueren einer ungültigen Region schneller geht: Benötigt dies ohne Reparaturfunktion mindestens zwei Generationen (Erstellen einer ungültigen Lösung, diese überlebt in die nächste Generation und erzeugt durch Kreuzung eine gültige Lösung auf der „anderen Seite“ des ungültigen Bereichs) kann dies mit Reperaturfunktion innerhalb einer einzigen Generation erfolgen (die erstellte ungültige Lösung wird durch Reparatur in eine gültige Lösung auf der „anderen Seite“ des ungültigen Bereichs umgewandelt). Zudem ist die Wahrscheinlichkeit eines solchen Vorgangs mit Reparaturfunktionen höher, da dabei nicht das Überleben einer (mit Strafe versehenen) ungültigen Lösung vorausgesetzt wird.

Wichtig ist allerdings bei der Verwendung von Reparaturfunktionen, dass diese so flexibel sind, dass sie das Durchqueren der ungültigen Bereiche auch wirklich zulassen und nicht eine ungültige Lösung immer sofort wieder dorthin zurück werfen, wo diese Lösung ihren Ursprung hatte. Eine solche Reparaturfunktion hätte den Effekt, dass bestimmte Regionen des Lösungsraumes gar nicht, oder nur sehr schwer zu erreichen sind.

Durch die Einführung einer flexibleren Reparaturfunktion lässt sich auch dem in Abschnitt 7.2.1.2 erwähnte Problem begegnen: In Abschnitt 5.3.3.4 wurde bereits angedeutet, dass die Liste der kritischen Bauwerke für die Matrix-Repräsentation in zufälliger Reihenfolge abgearbeitet werden sollte. Dies allein genügt jedoch nicht vollständig, um zu verhindern, dass bestimmte Lösungen bevorzugt werden. Wirklich frei von derartigen Einflüssen wird die Suche erst durch das Einführen eines weiteren Zufallsfaktors: Es entstehen unterschiedliche Lösungen, je nachdem ob man bei der Reparatur bei der Suche nach einem Tauschpartner für ein nicht oder zu spät im Terminplan vorkommendes kritisches Bauwerk vorne (also im Jahr 1) oder hinten (also im Jahr seiner Instandsetzungsfrist) anfängt. Trifft man diese Entscheidung zufällig, so weist die Reparatur keine Bevorzugung einzelner Lösungen mehr auf.

Zusammenfassend lässt sich sagen, dass die Kombination der drei in Abschnitt 5.3.3 beschriebenen Mechanismen zur Behandlung ungültiger Lösungen zu einem sehr guten Verhalten der Genetischen Algorithmen auf dem beschriebenen Problem führt.

	$\rho = 0.5$	$\rho = 0.4$	$\rho = 0.3$
Mittelwert [Fzgs*10^8]	6.3631	6.4243	6.2946
Beste Lösung [Fzgs*10^8]	6.1427	6.1483	6.1238
Standardabweichung [%]	3.16	2.81	2.72

Tabelle 7.3.: Ergebnisse aus jeweils zehn Testläufen mit *Elitist Ant System* über 50 Iterationen mit verschiedenen Verdunstunsfaktoren ρ auf Szenario 2. Die beste bekannte Lösung liegt bei $6.0526*10^8$ Fzgs (Fahrzeugsekunden), eine händisch konstruierte Lösung erreichte $6.0811 * 10^8$ Fzgs.

7.2.2. Ameisenalgorithmen

7.2.2.1. Elitist Ant System

Auf Szenario 2 wurden Testläufe des *Elitist Ant System* (Abschnitt 6.2.2) mit verschiedenen Werten für ρ (zwischen 0.3 und 0.5) durchgeführt (vgl. Tabelle 7.3). Unabhängig vom gewählten Wert für ρ zeigten sich jedoch die gleichen drei gravierenden Mängel dieses Algorithmus:

Zum einen brauchte der Algorithmus in allen Läufen einige Iterationen, um überhaupt gültige Lösungen zu finden. Dies dauert zwar nicht ganz so lange, wie bei den Genetischen Algorithmen ohne Reparaturfunktion (in der Regel wurden nach etwa fünf Iterationen erste gültige Lösungen gefunden), dennoch wird dabei einiges an Rechenzeit verschwendet. Diesem Effekt ließe sich zwar dadurch begegnen, dass der Einfluss der heuristischen Information η verstärkt wird, etwa dadurch, dass β erhöht wird, dies würde aber zu einer zu starken Beeinflussung in Richtung Greedy-Lösung und damit zu einer zu starken Einschränkung des Lösungsraumes führen.

Gravierender ist jedoch, dass die Existenz einer gültigen Elitist-Lösung nicht garantiert, dass auch in jedem Iterationsschritt eine gültige Lösung gefunden wird (vgl. auch Abbildung 7.8). In einem Extremfall führte dies sogar dazu, dass in einem Lauf über 100 Iterationen überhaupt nur im dritten Schritt eine gültige Lösung gefunden wurde, in den darauf folgenden Schritten nur noch ungültige Lösungen. Auch die Elitist-Lösung wurde kein zweites Mal gefunden. Der Einfluss der gültigen Elitist-Lösung war hier wohl zu schwach gegenüber den ungültigen Lösungen mit sehr guten Zielfunktionswerten.

Ein zu starker Einfluss der Elitist-Lösung führt jedoch zum dritten Mangel dieses Algorithmus: Dadurch, dass diese Lösung in jeder Iteration Pheromon ablagert, wird ihr Einfluss innerhalb weniger Iterationen so stark, dass die Suche vorzeitig auf diese Lösungen konvergiert. Dieser Effekt nimmt mit abnehmendem ρ zu, so dass in einzelnen Läufen mit $\rho = 0.3$ über 100 Iterationen so gut wie keine Verbesserung der Lösung stattfand. Wahrzunehmen war dieser Effekt jedoch bei allen Testläufen mit allen untersuchten Werten für ρ. Dies führt dazu, dass mit

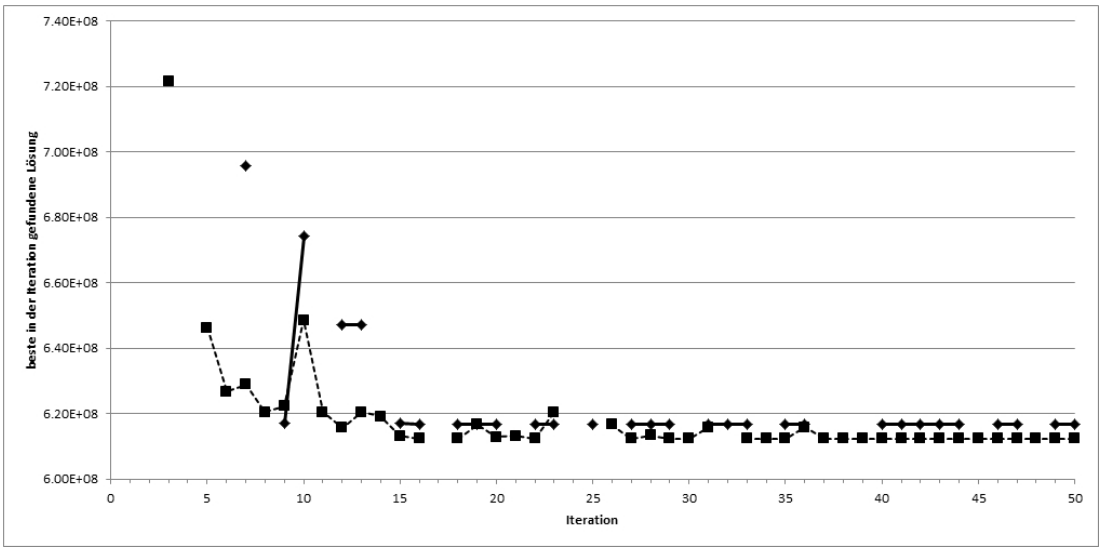

Abbildung 7.8.: Die jeweils besten pro Iteration gefundenen Lösungen in zwei Testläufen mit dem Elitist Ant System (die mit Rauten ◊ gekennzeichneten Lösungen stammen von einem Lauf mit $\rho = 0.5$, die mit Quadraten □ gekennzeichneten von einem Lauf mit $\rho = 0.3$) auf Szenario 2. Man kann leicht erkennen, dass in vielen Iterationen keine gültige Lösung gefunden wurde. Die erreichten Werte sind weit entfernt von der besten bekannten Lösung mit $6.0526 * 10^8$ Fzgs.

dem *Elitist Ant System* wesentlich weniger gute Lösungen gefunden wurden als mit allen anderen untersuchten Verfahren.

7.2.2.2. Rank-Based Ant System

Das Verhalten der Läufe mit *Rank-Based Ant System* sehr ähnlich dem von Läufen mit dem *Elitist Ant System*: Auch hier führt die starke Elitist-Lösung zu einer vorzeitigen Konvergenz auf weit vom Optimum entfernten Lösungen. Eine Lösung muss dazu hier im Gegensatz zum *Elitist Ant System* nicht die ganze Population übernehmen, es genügt die Übernahme der relativ kleinen Elitist-Gruppe der $(w - 1)$ besten Ameisen (vgl. Abbildung 7.9).

Dies suggeriert, dass die vorzeitige Konvergenz in diesem Fall noch eher eintritt als beim *Elitist Ant System*, doch das Gegenteil ist der Fall: In den Läufen mit dem *Rank-based Ant System* fand die Konvergenz einige Iterationsschritte später statt als in denen mit dem *Elitist Ant System*. Gleichzeitig waren die Lösungen, die dabei gefunden wurden, um einiges besser.

Erklären lässt sich dies wiederum über die Elitist-Gruppe: Da die in ihr enthaltenen Lösungen ebenfalls recht stark gewichtet werden, sind sie in einem gewissen Umfang dazu in der Lage, die Suche von der Umgebung der Elitist-Lösung wegzuführen und ihr so die Möglichkeit zu geben, auch andere Regionen zu durchsu-

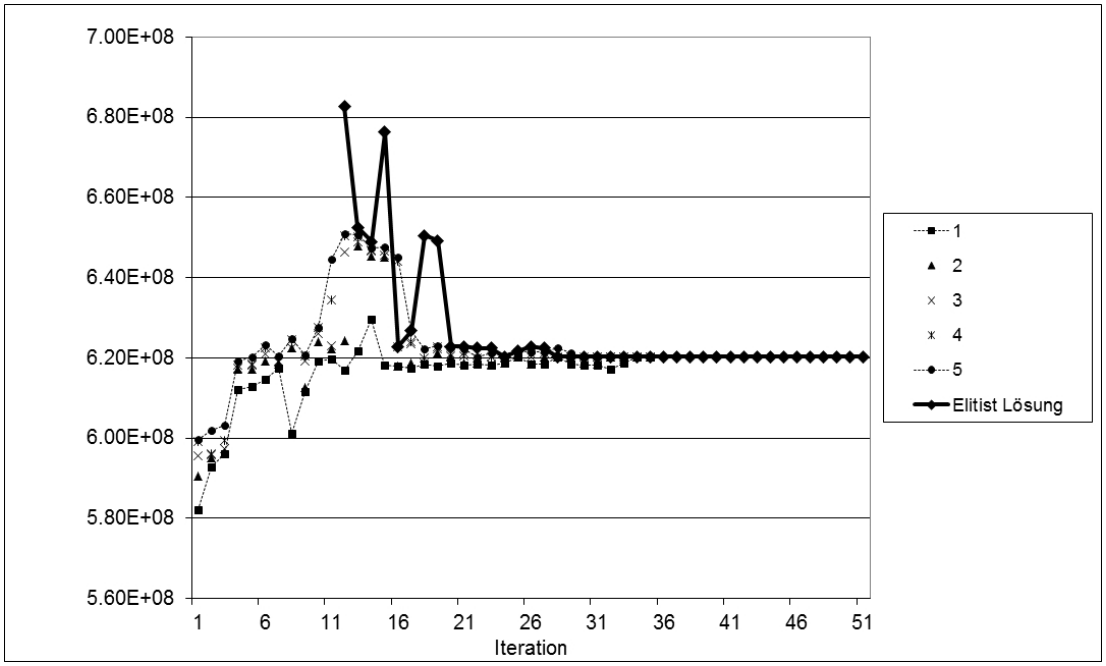

Abbildung 7.9.: Die Übernahme der Elitist-Gruppe durch eine relativ gute gültige Lösung in einem Testlauf auf Szenario 2. Die Ziffern 1 bis 5 bezeichnen die Ameisen mit den entsprechenden Rängen. Anfangs besteht die Elitist-Gruppe rein aus ungültigen Lösungen. Nach dem Auftreten der ersten gültigen Lösung dauert es nur wenige Iterationen, bis die Elitist Lösung Teil der Elitist-Gruppe wird. Nochmals wenige Iterationen später entsprechen alle Pheromon ablagernden Lösungen der Elitist Lösung.

	$\rho = 0.4$	$\rho = 0.3$	$\rho = 0.2$	$\rho = 0.4$, β dynamisch	$\rho = 0.4$, Elitist Ant dynamisch
Mittelwert [Fzgs*10^8]	6.2453	6.2825	6.1997	6.3006	6.1823
Beste Lösung [Fzgs*10^8]	6.1553	6.1232	6.1153	6.1641	6.1096
Standard-abweichung [%]	2.17	2.34	1.65	2.29	0.63

Tabelle 7.4.: Ergebnisse aus jeweils zehn Testläufen mit Rank-Based Ant System über 100 Iterationen mit verschiedenen Verdunstunsfaktoren ρ auf Szenario 2. In den letzten beiden Spalten aufgeführt sind die Ergenisse von Läufen mit dynamischem β bzw. dynamischer Elitist Ant (EA). Die beste bekannte Lösung liegt bei $6.0526 * 10^8$ Fzgs (Fahrzeugsekunden), eine händisch konstruierte Lösung erreichte $6.0811 * 10^8$ Fzgs.

chen. Da in der Elitist-Gruppe auch ungültige Lösungen enthalten sein können, können auch ungültige Regionen leichter durchquert werden als beim *Elitist Ant System*.

Dieses Verhalten zeigt sich auch darin, dass in Testläufen mit verschiedenen Verdunstungsfaktoren ρ zwischen 0.2 und 0.4, im Gegensatz zum *Elitist Ant System*, mit abnehmendem ρ eine Verbesserung der Ergebnisse eintritt (vgl. Tabelle 7.4). So zeigen die Läufe mit $\rho = 0.4$ eine sehr frühe Konvergenz zu sehr schlechten Lösungen und in einzelnen Läufen so gut wie keine Verbesserung über die Iterationen, während in den Läufen mit $\rho = 0.2$ noch in späteren Generationen Verbesserungen der Lösung festzustellen waren und damit in einzelnen Läufen Lösungen erreicht wurden, die an die von den Genetischen Algorithmen gefundenen Lösungen heranreichen. Zu erklären ist dies damit, dass mit zunehmenden ρ Lösungen der Elitist-Gruppe einer früheren Iteration schneller „vergessen“ werden während sie mit niedrigerem ρ noch länger zur Verfügung stehen, und so den Einfluss der aktuellen Elitist-Lösung reduzieren. Allerdings nimmt der Einfluss der Elitist-Lösung auch mit einem geringen ρ innerhalb einiger Iterationen so stark überhand, dass die Lösungen auch hier im Schnitt schlechter ausfallen als bei den Läufen mit Genetischen Algorithmen.

Versuche, dieses Verhalten zu umgehen, etwa durch die Reduzierung des Werts für β über die Generationen (vgl. Abschnitt 6.3.3) oder durch die Einführung einer dynamischen *Elitist Ant* (vgl. Abschnitt 6.2.2), zeigen nicht den gewünschten Effekt. Die vorzeitige Konvergenz tritt zu schnell ein, als dass eine Abnahme von β nach Gleichung 6.17 einen nennenswerten Einfluss auf die Routenwahl haben könnte. Ein Wechsel der *Elitist Ant* spätestens alle 5 Generationen zeigt eine leichte Verbesserung des Ergebnisses: Die neue Elitist-Lösung ist allerdings meistens zu schwach um sich durchzusetzen, so dass die Suche im größten Teil

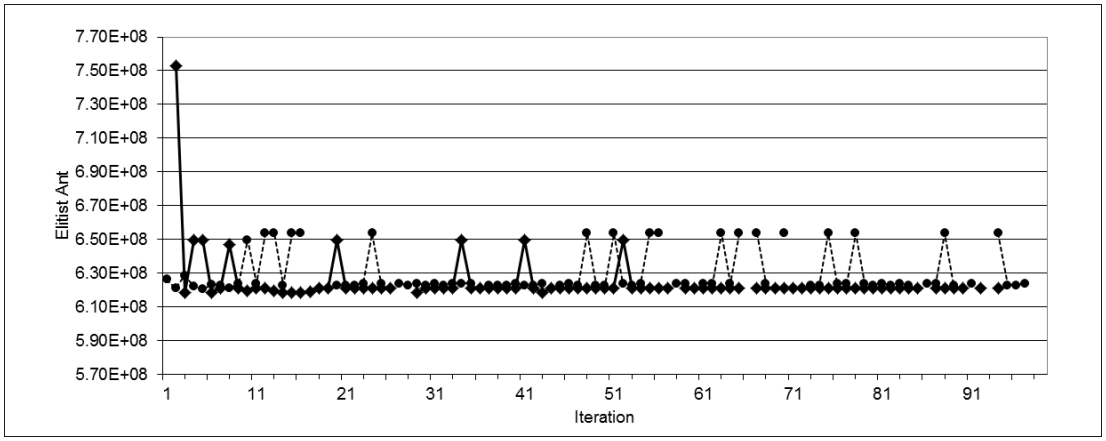

Abbildung 7.10.: Die Zielfunktionswerte der Elitist Ant über jeweils 100 Iterationen in zwei Läufen des Rank-Based Ant System mit dynamischer *Elitist Ant* ($\rho = 0.4$, $m = 100$). Wie zu erkennen ist, schafft es der erzwungene Wechsel der *Elitist Ant* nach spätestens fünf Iterationen nicht, die Suche von der alten Elitist-Lösung wegzuführen. Meistens kehrt die Suche schon in der nächsten Iteration zu dieser zurück.

der Fälle schon im nächsten Iterationschritt nach einem Wechsel zur alten Elitist-Lösung zurückkehrt (vgl. Abbildung 7.10). In manchen Fällen führt der Wechsel sogar dazu, dass die Suche wieder in die Nähe einer Lösung zurückkehrt, die vor mehreren Generationen Elitist-Lösung war und deutlich schlechter ist, als die bisher beste Lösung des Laufs. Dennoch zeigt sich in den Läufen mit dynamischer *Elitist Ant* eine leichte Verbesserung der Ergebnisse.

7.2.2.3. Ant Colony System

Ebenfalls auf dem Szenario 2 wurden Testläufe mit dem *Ant Colony System* durchgeführt (siehe Tabelle 7.5). Zunächst wurde η, wie für die anderen Ameisenalgorithmen auch, einfach abhängig vom Abstand von der Instandsetzungsfrist definiert (vgl. Abschnitt 6.3.2).

Bei den Testläufen mit $\rho = 0.1$ und $m = 10$ (entsprechend der Empfehlungen aus Dorigo & Gambardella (1997)), zeigte sich das Problem der vorzeitigen Konvergenz behoben: Auch in späten Iterationen (in Läufen mit 100 Iterationen bis zur letzten Iteration) gibt es noch Änderungen in der Lösungsgüte. Allerdings findet die Verbesserung der Lösung über die Iterationen im Vergleich zu den untersuchten Genetischen Algorithmen sehr langsam statt, so dass die Lösungen nach 100 Iterationen gefundenen Lösungen im Schnitt immer noch schlechter sind, als die nach 100 Generationen Genetischem Algorithmus erreichten. Dies liegt zum Teil darin begründet, dass in vielen Iterationen gar keine gültige Lösung gefunden wird und dass gute Lösungen immer wieder verloren gehen, so dass die Suche zu schlechteren Lösungen „zurückspringt“.

	$m = 10$, $\rho = 0.1$	$m = 5$, $\rho = 0.2$	$m = 10$, $\rho = 0.4$	$m = 10$, $\rho = 0.1$, angepasstes η
Mittelwert [Fzgs*10^8]	6.3163	6.2802	6.2489	6.1074
Beste Lösung [Fzgs*10^8]	6.1712	6.1191	6.1243	6.0968
Standard-abweichung [%]	2.32	2.37	2.43	0.11

Tabelle 7.5.: Ergebnisse aus jeweils zehn Testläufen mit Ant Colony System über 100 Iterationen auf Szenario 2. In den ersten beiden Spalten dargestellt sind Läufe mit nach Dorigo & Gambardella (1997) aufeinander abgestimmtem m und ρ. In der dritten Spalte wurde ein zu großer Verdunstungsfaktor ρ für die Größe der Ameisenkolonie m gewählt. Die letzte Spalte zeigt Ergebnisse mit Läufen, in denen ein Wert für η verwendet wurde, der beide Randbedingungen berücksichtigt. Die beste bekannte Lösung liegt bei $6.0526 * 10^8$ Fzgs (Fahrzeugsekunden), eine händisch konstruierte Lösung erreichte $6.0811 * 10^8$ Fzgs.

Erhöht man ρ unter gleichzeitigem Zurücksetzen der Ameisenanzahl auf $m = 5$ auf 0.2, dann werden, aufgrund der geringen Anzahl an Ameisen noch seltener gültige Lösungen gefunden (in einem Lauf wurde erst nach 45 Iterationen die erste gültige Lösung entdeckt), diese setzen sich damit auch wesentlich stärker durch, was wiederum zu einer weiteren Verlangsamung der Suche führt.

Wird die Anzahl der Ameisen zu groß für das verwendete ρ gewählt, wird der Einfluss der Elitist-Lösung wiederum zu stark. Testläufe auf dem Szenario 2 mit $\rho = 0.4$ bei $m = 5$ Ameisen über 2000 Iterationen zeigten, dass die Suche damit ihre Stabilität verliert. Nach anfänglich gutem Konvergenzverhalten fangen die Lösungen an zu oszillieren. Die Suche springt wahllos in verschiedene Bereiche des Suchraums, gute Lösungen gehen somit verloren (vgl. Abbildung 7.11).

Eine wesentliche Verbesserung lässt sich mit der in Abschnitt 6.3.2 beschriebenen Anpassung von η, so dass neben der Sicherheitsrandbedingung auch die Budgetrandbedingung in die Entscheidung der Ameisen eingeht, erreichen. Durch das strenge Entscheidungskriterium des *Ant Colony Systems*, bei dem in 90 Prozent (q_0) der Fälle die Lösung mit dem höchsten Produkt aus τ und η gewählt wird, wird dafür gesorgt, dass schon in der ersten Iteration vornehmlich gültige Lösungen gefunden werden. Die Vorteile des *Ant Colony Systems*, das durch die Reduktion von Pheromon auf gewählten Knoten eine vorzeitige Konvergenz verhindert, bleiben dabei weiter bestehen.

Auf diese Weise können bei gleicher Anzahl von Iterationen (bzw. Generationen) Ergebnisse erreicht werden, die fast genau so gut sind, wie die durch Genetische Algorithmen gefundenen Lösungen. Da das *Ant Colony System* jedoch mit nur 10 Ameisen arbeitet, während bei den Genetischen Algorithmen Populationsgrößen

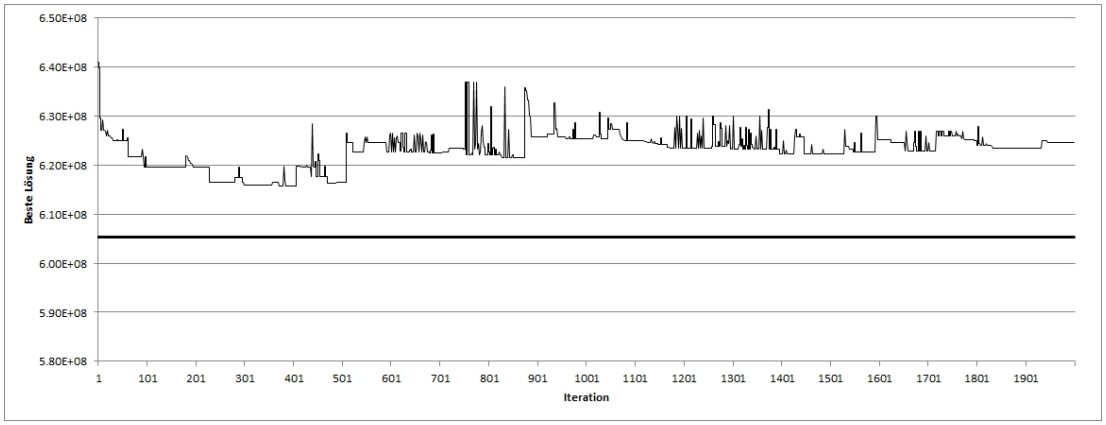

Abbildung 7.11.: Testlauf mit $\rho = \xi = 0.4$ und $m = 5$ über 2000 Generationen. Durch die fehelnde Ausgeglichenheit zwischen entferntem und abgelagetem Pheromon geht der Suche die Stabilität verloren. Die Suche springt willkürlich durch den Suchraum, gute Lösungen gehen verloren ohne zur Steuerung der Suche genutzt zu werden. Die dicke durchgezogene Linie markiert die beste bekannte Lösung für Szenario 2.

von 50 oder gar 100 verwendet werden, kommt dieses dabei mit wesentlich weniger Auswertungen der Zielfunktion aus. Nachdem die Verkehrssimulation der zeitintensivste Vorgang der Optimierung ist, bedeutet dies eine gravierende Einsparung an Rechenzeit gegenüber den Genetischen Algorithmen.

7.2.2.4. Lokale Suche

In Abschnitt 6.3.5 wurde die Möglichkeit der Kombination der Ameisenalgorithmen mit einer lokalen Suche erwähnt. Wie dort schon ausgeführt, bedeutet diese auch einen höheren Rechenaufwand, da jede bei der lokalen Suche erzeugte Nachbarlösung ebenfalls mit dem Verkehrssimulator ausgewertet werden muss. Aus diesem Grund kann nur ein begrenzter Teil der Nachbarschaft einer Lösung überhaupt untersucht werden (vgl. Abschnitt 6.3.5 und Abbildung 6.10 zur Größe der Nachbarschaft). Die Wahrscheinlichkeit, dass bei einer derartigen zufälligen Wahl tatsächlich eine bessere Lösung gefunden wird, ist sehr gering.

Testläufe auf dem Szenario 2, bei denen das *Ant Colony System* mit einer lokalen Suche gekoppelt wurde, zeigten noch eine weitere Schwierigkeit: Da bei dem in Abschnitt 6.3.5 beschriebenen Algorithmus zur lokalen Suche nur die Sicherheitsrandbedingung nicht aber die Budgetrandbedingung berücksichtigt wird, wird durch die lokale Suche eine gültige Lösung oftmals in eine ungültige Lösung umgewandelt. Dies führt dazu, dass in den Testläufen nur in einem geringen Bruchteil der Iterationen gültige Lösungen gefunden wurden. Damit fand auch über die Iterationen nur eine geringe Verbesserung der Lösungen statt.

Aus den hier beschriebenen Gründen ist von einer Kopplung mit einer lokalen Suche eher abzuraten.

7.2.3. Zusammenfassung

Eine Reihe von Testläufen auf den verschiedenen Testszenarien wurde durchgeführt, um die Eignung der einzelnen in den Abschnitten 5.3 und 6.3 beschriebenen Algorithmen für das Instandsetzungsproblem zu untersuchen. Dabei ging es sowohl um die grundsätzliche Eignung der Algorithmen als auch darum, zu testen, wie die Lösungsgüte durch verschiedene Faktoren beeinflusst werden kann.

Prinzipiell lässt sich feststellen, dass zwei der drei untersuchten Dialekte von Ameisenalgorithmen, das *Elitist Ant System* und das *Rank-Based Ant System*, für diese Problemstruktur sehr schlecht geeignet sind. Auch Modifikationen am Algorithmus wie eine dynamische *Elitist Ant* oder eine dynamische Wichtung der heuristischen Information η waren nicht in der Lage, dies zu verbessern.

Genetische Algorithmen und das *Ant Colony System* schnitten in den Test dagegen sehr gut ab. Für die Qualität der Genetischen Algorithmen von Bedeutung ist dabei vor allem die Wahl geeigneter und aufeinander abgestimmter Operatoren für Kreuzung und Mutation. Ebenfalls von Bedeutung sind die gewählten Selektionsmechanismen: Nur wenn auch ungültige und schlechte Lösungen in die nächste Generation übernommen werden und diese auch die Möglichkeit haben, sich fortzupfanzen, ist dauerhaft eine Verbesserung der Lösungen festzustellen. Andererseits dürfen einmal gefundene gute Lösungen der Population nicht verloren gehen. Ein Selektionsmechanismus, der beide Anforderungen erfüllt, ist die Turnierselektion mit Mehrfachauswahl. Die gewählte Repräsentation hat in den hier durchgeführten Versuchen nur einen sehr geringen Einfluss auf die Lösungsgüte, ein Hinweis darauf, dass beide hier untersuchten Repräsentationen gut an die Problemstruktur angepasst sind.

Das *Ant Colony System* zeigte bei aufeinander abgestimmten Werten für ρ und m gute Ergebnisse. Diese lassen sich noch weiter dadurch verbessern, dass in die heuristische Information η basierend auf beiden Randbedingungen bestimmt wird. Mit dieser Anpassung erreicht auch das *Ant Colony System* sehr gute Lösungen.

Die Qualität der mit dem *Ant Colony System* gefundenen Lösungen liegt nach gleicher Generationenzahl zwar leicht unter der der mit Genetischen Algorithmen gefundenen Lösungen[5]. Dafür werden sie in einem Bruchteil der Zeit erreicht [6]. Daher empfiehlt sich für die Verwendung in der Praxis eher das *Ant Colony*

[5] In den Testläufen auf Szenario 2 hatten die mit Ant Colony System gefundenen Lösungen im Schnitt 0.7% höhere Zielfunktionswerte als die mit einem Genetischen Algorithmus mit Matrix-Repräsentation gefundenen Lösungen.

[6] Bei $m = 10$ benötigt das Ant Colony System pro Iteration nur ein Fünftel der Zielfunktionsauswertungen gegenüber einem Genetischen Algorithmus mit $\mu = 50$ und damit auch etwa ein Fünftel der Zeit.

System. Prinzipiell eignen sich jedoch beide Algorithmen gut zur Lösung des Instandsetzungsproblems.

Testläufe auf Szenario 4 bestätigten die auf den anderen Szenarien gemachten Beobachtungen. Die besten Lösungen mit $1.2084 * 10^9$Fzgs wurden auch hier mit Genetischen Algorithmen gefunden. Dies bedeutet eine Einsparung von 1.4% gegenüber einer durch frühestmögliche Terminierung der Bauwerke erzeugten Greedy-Lösung. Doch auch die mit dem Ant Colony System gefundenen Lösungen lieferten noch eine Einsparung von 1.2% gegenüber dieser Greedy Lösung. Dies zeigt, dass diese Algorithmen auch auf realen Szenarien sinnvoll arbeiten. Auch die zuvor beschriebenen Unzulänglichkeiten der anderen untersuchten Ameisenalgorithmen ließen sich auf dem Szenario 4 beobachten.

7.3. Mehrkriterielles Problem

Alle Testläufe zur mehrkriteriellen Optimierung fanden auf Szenario 2 statt. Um die Qualität der Ergebnisse der einzelnen Läufe beurteilen zu können, wurde für jedes Zielkriterium der jeweils beste Wert über eine einkriterielle Optimierung bestimmt. Des weiteren wurde die Menge der nicht-dominierten Lösungen aus allen Lösungen bestimmt, die bei den verschiedenen Testläufen gefunden wurden. Diese Punkte stellen damit die bekannten Lösungen dar, die am nächsten an die tatsächliche Pareto-Front angenähert sind (siehe Abbildung 7.12).

Bei allen Testläufen zeigte sich jedoch, dass das Szenario 2 bezogen auf das mehrkriterielle Optimierungsproblem sehr schlecht konditioniert ist: Das Optimum für das Zielkriterium der Gruppierung von Bauwerken (Dep[7]) liegt bei 0 (also können für alle Gruppen die Bauwerke, die jeweils zu ihnen gehören, zusammen instand gesetzt oder aber ganz aus dem Terminplan entfernt werden). Ein Großteil der Lösungen, die dieses Optimum erreichen, ist jedoch ungültig. Die wenigen gültigen Lösungen, die in diesem Bereich liegen, sind damit nur sehr schwer zu erreichen. Daher wurden diese Lösungen in den Testläufen nur sehr selten gefunden. Ob dies der Fall war, hing dabei nicht von dem verwendeten Algorithmus ab, sondern allein von der Zusammensetzung der Startgeneration. Enthielt diese bereits Lösungen mit niedrigen Werten für Dep, so wurden auch weiterhin Lösungen in diesem Bereich gefunden. Waren die Werte für Dep für alle Lösungen der Startgeneration dagegen hoch, so war es mit keinem der untersuchten Algorithmen möglich, Lösungen im Bereich des Optimums für dieses Kriterium zu erreichen.

[7] Abkürzung für engl. *dependencies*: Abhängigkeiten

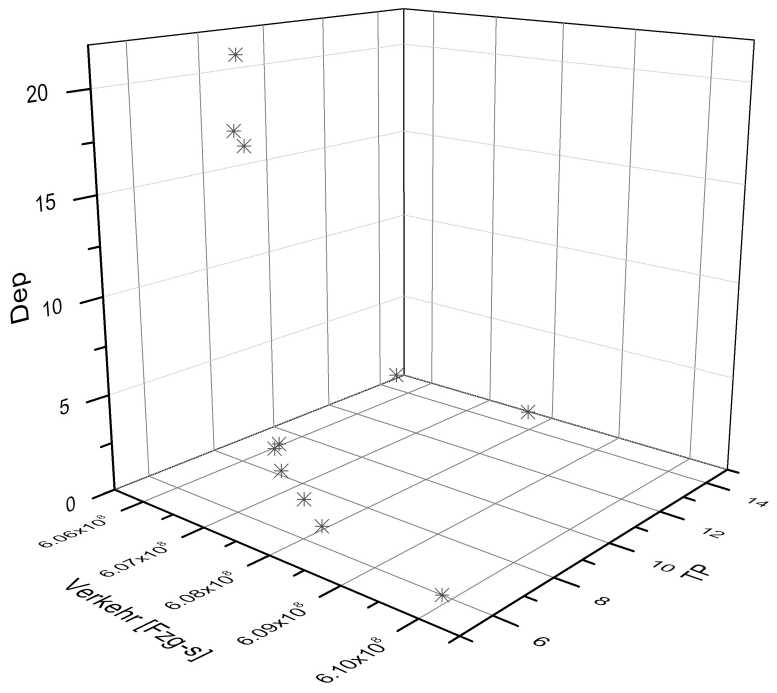

Abbildung 7.12.: Die nicht-dominierten Lösungen aus allen Läufen auf dem mehrkriteriellen Szenario 2. Verkehr steht für die Ergebnisse der Verkehrssimulation in Fzgs. Dep ist das Maß für das Zielkriterium der Gruppierung von Bauwerken. Aufgetragen ist hier die Summe der Abweichung einzelner Bauwerke vom Gruppenmaximum. TP ist das Zielkriterium der Synergien mit eine dritten Partei. Gemessen ist dieses Zielkriterium als die Summe der Abweichungen zwischen den tatsächlichen und den von dritter Seite geplanten Terminen.

7.3.1. Genetische Algorithmen

7.3.1.1. Non-Generational Genetic Algorithm for Multi-objective Optimization

Wie in Abschnitt 5.4.3 erwähnt, muss in der Startpopulation für den mehrkriteriellen Genetischen Algorithmus ein breites Spektrum an gültigen Lösungen zur Verfügung stehen, um so sicher zu stellen, dass nicht nur ein kleiner Bereich der Pareto-Front abgesucht wird. Zu diesem Zweck wurde im mehrkriteriellen Fall, genau so wie im einkriteriellen Fall, ein einkriterieller Lauf zur Minimierung der Randbedingungsverletzungen vorgeschaltet. Mit dieser wird eine Startpopulation aus gültigen Lösungen erstellt.

Besteht beim NGGA-MO jedoch die gesamte Startpopulation aus gültigen Lösungen, ergibt sich daraus ein neues Problem: Auch wenn die Startpopulation ausreichend Diversität aufweist, spaltet sich die Bevölkerung innerhalb weniger Generationen in einige wenige Populationsinseln auf, Lösungen außerhalb dieser Inseln werden kaum noch gefunden, eine weitere Annäherung an die Pareto-Front ist damit nicht mehr gegeben (vgl. Abbildung 7.13).

Der Grund liegt auch hier in der Vernachlässigung der ungültigen Lösungen: Da sie von allen Lösungen der Population dominiert werden, unabhängig von ihren eigentlichen Zielfunktionswerten, weisen sie gleich nach ihrer Erzeugung sehr schlechte Fitnesswerte auf. Damit überleben sie nur sehr wenige Generationen und werden mit einer gegen null gehenden Wahrscheinlichkeit als Eltern ausgewählt. Um die ungültigen Regionen des Lösungsraumes zu durchqueren, ist es jedoch, wie schon mehrmals erwähnt, nötig, dass auch ungültige Individuen die Möglichkeit zur Fortpflanzung bekommen.

Eine Möglichkeit dies zu umgehen, besteht darin, in die Startpopulation auch ungültige Lösungen aufzunehmen. Testläufe mit einer zur Hälfte gültigen, zur Hälfte ungültigen Startpopulation zeigten jedoch, dass sich das Problem auf diese Weise nicht beheben lässt (vgl. Abbildung 7.14): Während der ersten Generationen wurden die ungültigen Lösungen der Startlösung nach und nach eliminiert, später fand die gleiche Aufspaltung in Inseln statt wie bei einer rein gültigen Startpopulation. Über alle Testläufe gesehen schnitten diejenigen mit einer gemischten Startpopulation sogar schlechter ab.

Die Nischengröße σ wurde in allen Testläufen so gewählt, dass zwei Lösungen in der gleichen Nische liegen, wenn sie sich nur in zwei oder weniger Einträgen unterscheiden. Versuche über ein erhöhtes c_ω die Nischenfüllung stärker zu wichten zeigten ebenfalls nicht den gewünschten Effekt einer diversen Population. Selbst mit $c_\omega = c_\delta = 1$ bildeten sich Inseln und die Suche konzentrierte sich vornehmlich auf die Eliminierung ungültiger Lösungen (vgl. Abbildung 7.15).

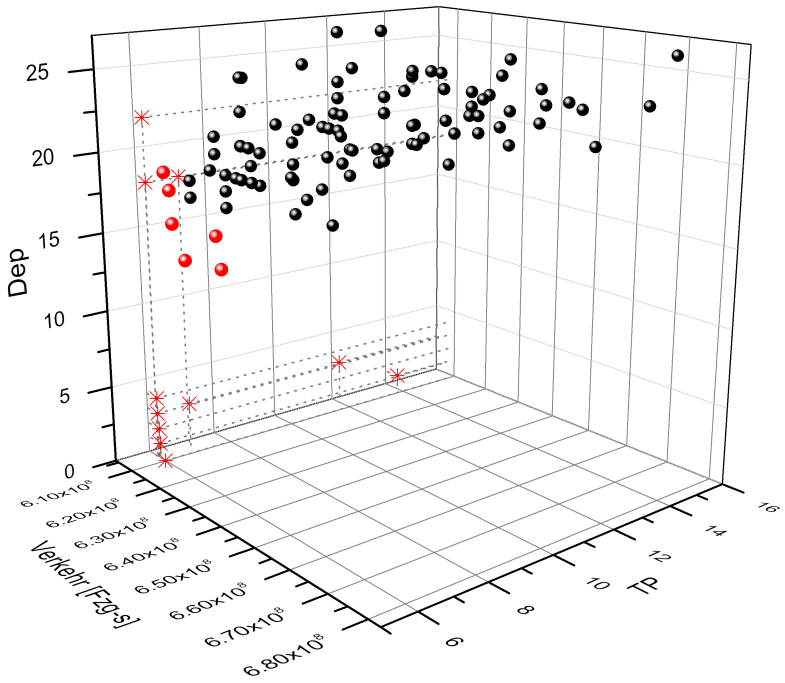

Abbildung 7.13.: Lösungen aus einem Lauf des NGGA-MO mit $c_\delta = 5$, $c_\omega = 0.1$, $n_1 = 10$ und $n_2 = 20$. Die schwarzen Kugeln markieren die Lösungen der Startgeneration, die roten die nicht-dominierten Lösungen nach 2000 Generationen. Die roten Sterne zeigen die nicht-dominierten Lösungen aus allen Testläufen (vgl. Abschnitt 7.3). Die nicht-dominierten Lösungen der letzten Generation liegen sehr dicht beieinander, die Suche beschränkte sich auf einen kleinen Insel-Bereich der Pareto-Front.

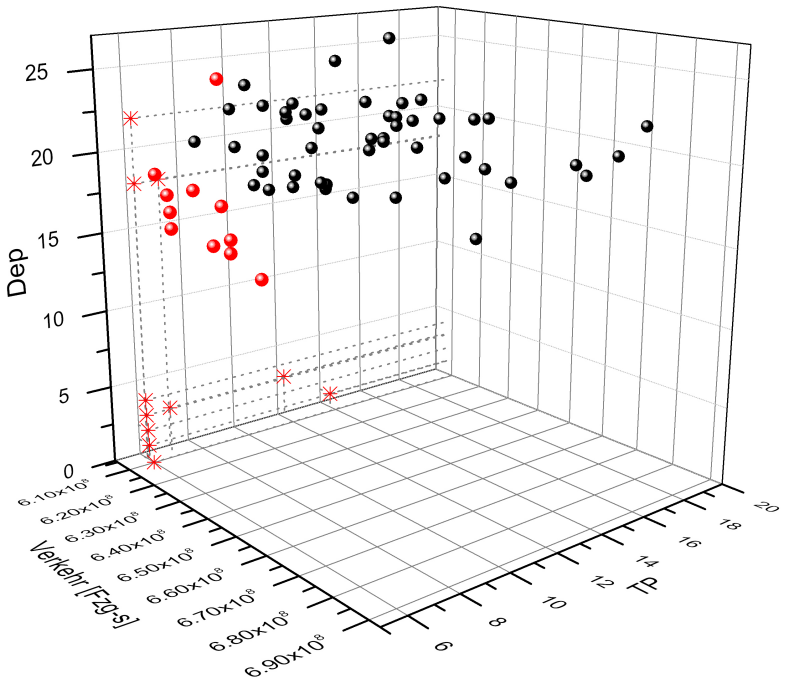

Abbildung 7.14.: Lösungen aus einem Lauf des NGGA-MO mit $c_\delta = 10$, $c_\omega = 0.05$, $n_1 = 5$ und $n_2 = 10$ mit einer Startgeneration, die zur Hälfte aus ungültigen Lösungen besteht. Die schwarzen Kugeln markieren die gültigen Lösungen der Startgeneration, die roten die nicht-dominierten Lösungen nach 2000 Generationen. Die roten Sterne zeigen die nicht-dominierten Lösungen aus allen Testläufen (vgl. Abschnitt 7.3). Auch hier beschränken sich die nicht-dominierten Lösungen der letzten Generation nur auf einen kleinen Bereich.

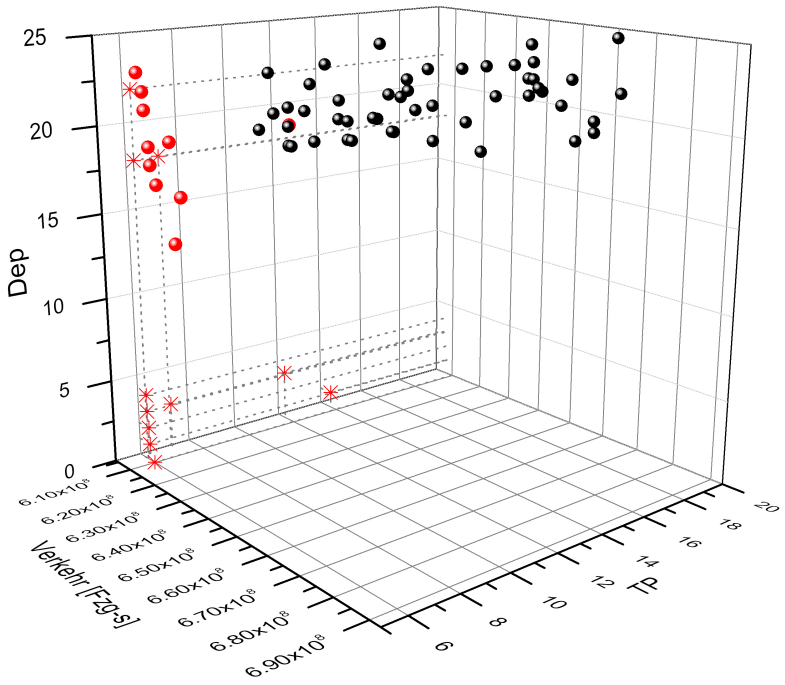

Abbildung 7.15.: Lösungen aus einem Lauf des NGGA-MO mit $c_\delta = 1$, $c_\omega = 1$, $n_1 = 10$ und $n_2 = 20$ mit einer Startgeneration, die zur Hälfte aus ungültigen Lösungen besteht. Die schwarzen Kugeln markieren die gültigen Lösungen der Startgeneration, die roten die nicht-dominierten Lösungen nach 2000 Generationen. Die roten Sterne zeigen die nicht-dominierten Lösungen aus allen Testläufen (vgl. Abschnitt 7.3). Die Bildung von Inseln ist auch hier zu erkennen.

7.3.1.2. Ansatz von Jaszkiewicz

Um den MOGA-Jas auf dem mehrkriteriellen Instandsetzungsproblem sinnvoll anwenden zu können, mussten Änderungen an diesem Algorithmus vorgenommen werden: In ersten Testläufen erwies sich das dynamische Festlegen der Größe der Startpopulation S als unpraktikabel, da S nach der Heuristik von Jaszkiewicz (2002) übermäßig groß sein kann (vgl. Abschnitt 5.4.2). Daher wurde für die weiteren Testläufe eine maximale Populationsgröße von 100 Individuen für die Startgeneration gewählt.

Eine zweite Änderung ist nötig, da in der hier vorgestellten Implementierung auf die im eigentlichen MOGA-LS von Jaszkiewicz (2002) vorgesehene lokale Suche verzichtet wird. Damit sinkt die Wahrscheinlichkeit, dass das durch Kreuzung neu entstandene Individuum entsprechend der gewichteten Fitnessfunktion besser ist als alle Individuen in der Elternmenge TP. Dies führt nach den Regeln des Algorithmus dazu, dass kaum neue Individuen in die Population aufgenommen werden. Dies kann wiederum dazu führen, dass die Suche stagniert. Andererseits könnten diese neuen Individuen die Population durchaus bereichern, da sie eventuell auf einer anders gewichteten Fitnessfunktion bessere Werte liefern, als der Rest der Population. Aus diesem Grund wurde der Algorithmus so abgeändert, dass jedes durch Kreuzung entstandene Individuum in die Gesamtpopulation aufgenommen wird.

Testläufe wurden mit verschiedenen Werten für K, also der Größe der Elternmenge TP, durchgeführt. Dabei zeigte sich bei der Wahl eines zu großen K, z.B. 8, die Tendenz, dass nur Lösungen gefunden wurden, die mittlere Werte für alle Zielfunktionen liefern, Lösungen mit Extremwerten traten dagegen nicht auf. Die nicht-dominierten Lösungen der letzten Generation liegen damit zwar nah an der Pareto-Front, allerdings decken sie diese nur unzureichend ab, sondern liegen sehr nah zusammen (vgl. Abbildung 7.16). Für ein $K = 4$ ergibt sich dagegen eine wesentlich stärkere Diversität der Lösungen und damit eine bessere Abdeckung der angenäherten Pareto-Front (vgl. Abbildung 7.17). Der Grund dafür liegt darin, dass bei der elitäreren Auswahl der Elternmenge mit kleinerem K Extremwerte stärker berücksichtigt werden.

Zusammenfassend lässt sich sagen, dass sich der MOGA-Jas bei geeigneter Wahl von K dem NGGA-MO überlegen zeigt, da die durch ihn gefundenen Lösungen eine wesentlich stärkere Diversität aufweisen. Die Testläufe weisen darauf hin, dass ein $K = 4$ für Probleminstanzen des hier behandelten Problemtyps im allgemeinen die beste Wahl ist. Als Nachteil muss jedoch bemerkt werden, dass die meisten Läufen mit MOGA-Jas in der letzten Generation nur sehr wenige nicht-dominierte Lösungen aufwiesen (in manchen Läufen waren es nur zwei oder drei). Damit erweist sich auch dieser Algorithmus als wenig geeignet für eine praktische Anwendung.

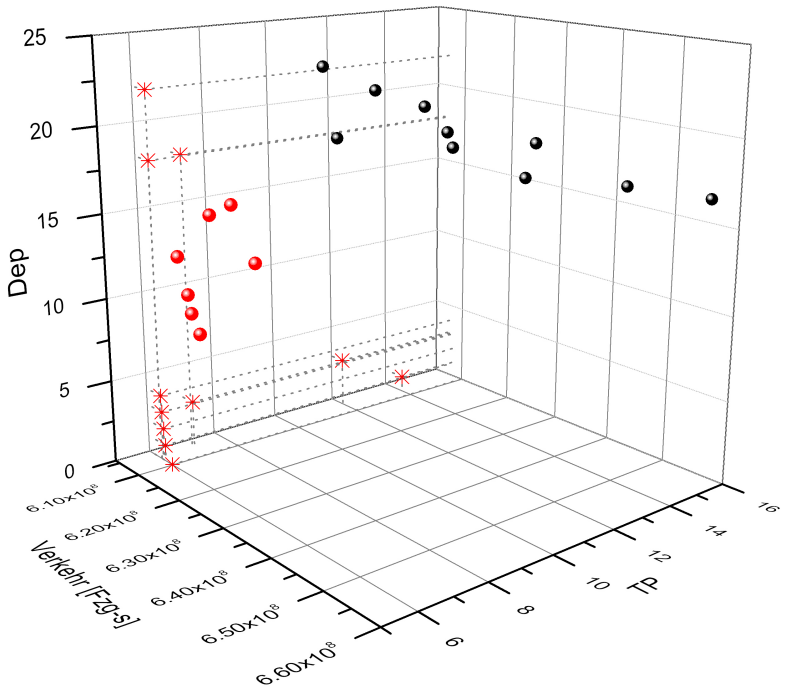

Abbildung 7.16.: Lösungen aus einem Lauf des MOGA-Jas mit $K = 8$. Die schwarzen Kugeln markieren die nicht-dominierten Lösungen der Startgeneration, die roten die nicht-dominierten Lösungen nach 2000 Generationen. Die roten Sterne zeigen die nicht-dominierten Lösungen aus allen Testläufen (vgl. Abschnitt 7.3). Die Lösungen der letzten Generation liegen sehr dicht beieinander bei mittleren Zielfunktionswerten. Lösungen mit Extremwerten treten dagegen nicht auf.

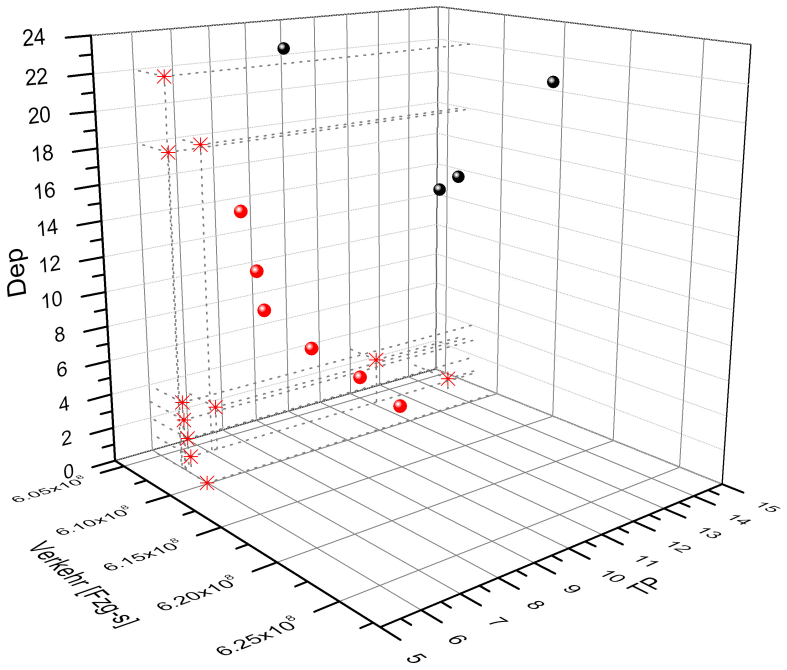

Abbildung 7.17.: Lösungen aus einem Lauf des MOGA-Jas mit $K = 4$. Die schwarzen Kugeln markieren die nicht-dominierten Lösungen der Startgeneration, die roten die nicht-dominierten Lösungen nach 2000 Generationen. Die roten Sterne zeigen die nicht-dominierten Lösungen aus allen Testläufen (vgl. Abschnitt 7.3). Die Lösungen der letzten Generation zeigen eine wesentlich stärkere Diversität als bei dem Lauf mit $K = 8$ aus Abbildung 7.16. Die angenäherte Pareto-Front wird besser abgedeckt.

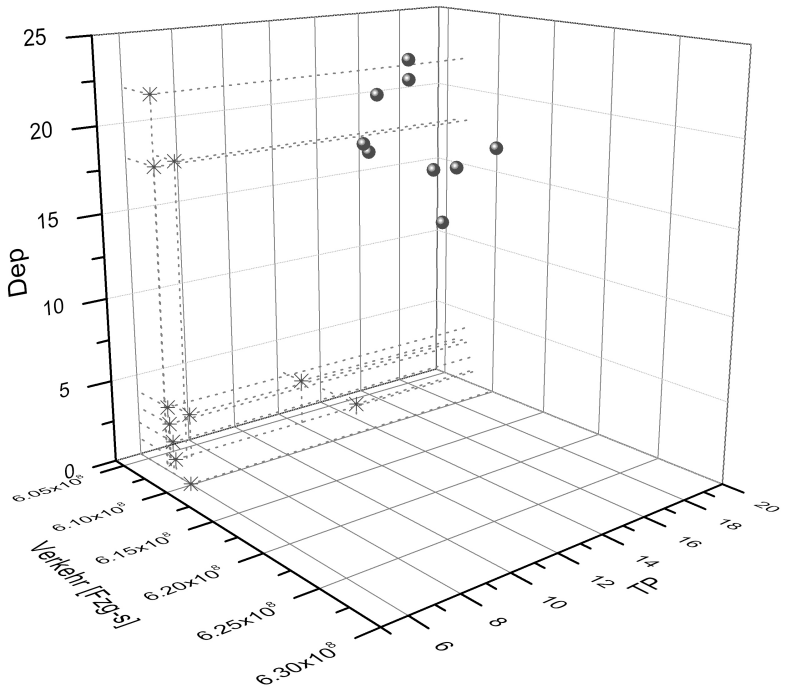

Abbildung 7.18.: Die nicht-dominierten Lösungen nach 100 Iterationen mit dem Bi-Criterion Ant System. Die durch den Algorithmus gefundenen Lösungen sind weit weg von den mit einem Stern markierten nicht-dominierten Lösungen aus allen Testläufen (vgl. Abschnitt 7.3).

7.3.2. Ameisenalgorithmen

7.3.2.1. Bi-Criterion Ant System

Das Bi-Criterion Ant System hat generell Schwierigkeiten, gültige Lösungen zu finden. In Testläufen dauerte es bis zu zehn Iterationen bis die erste gültige Lösung auftrat. Auch im weiteren Verlauf der Suche werden nur sehr selten neue nicht-dominierte Lösungen gefunden. Die Annäherung an die Pareto-Front geht daher nur sehr langsam vor sich.

Da das Bi-Criterion Ant System der einzige der untersuchten Algorithmen zur mehrkriteriellen Optimierung ist, der in jeder Iteration eine große Zahl von Lösungen auswerten muss (in den Testläufen wurden 50 Ameisen verwendet), wird dort pro Iteration auch wesentlich mehr Rechenzeit benötigt. Dies führt dazu, dass in der gleichen Zeit auch weniger Iterationen durchgeführt werden können.

Zusammen bewirkt dies, dass in einer für die Praxis akzeptablen Rechenzeit von circa 24 Stunden, das entspricht etwa 100 Iterationen, keine guten Ergebnisse erreicht werden können (vgl. Abbildung 7.18).

7.3.2.2. Pareto Ant Colony Optimization

PACO benötigt wie das Ant Colony System im einkriteriellen Fall nur wenige Ameisen. Die Testläufe wurden mit $\rho = 0.1$ und 10 Ameisen durchgeführt. Damit sind nur 10 Auswertungen der einzelnen Zielfunktionen pro Iteration nötig.

Durch das strenge Auswahlkriterium ist, wie auch beim Ant Colony System, zudem sicher gestellt, dass der Großteil der erstellten Lösungen gültig ist. Damit ist eine schnelle Annäherung der Suche an die Pareto-Front möglich. Innerhalb der 200 Iterationen der Testläufe wurde eine große Menge gültiger nicht-dominierter Lösungen gefunden.

Diese Lösungen zeichnen sich zudem durch eine hohe Diversität aus (Abbildung 7.19). Gegenüber den Genetischen Algorithmen haben die Ameisen dabei den Vorteil, dass die neuen Lösungen in jeder Iteration von Grund auf neu konstruiert werden, während die Genetischen Algorithmen immer aus vorhandenen Gen-Bausteinen schöpfen müssen. Gerade in Lösungsräumen mit wenigen und nicht zusammenhängenden gültigen Lösungen befähigt dies die Ameisen dazu, weitere Bereiche zu durchsuchen, während die Genetischen Algorithmen ihre Suche bald auf einzelne Inseln beschränken.

Die Wahl des Parameters q_0, also das Maß für die Strenge des Auswahlkriteriums, hat dabei, wie von Doerner *et al.* (2004) vorhergesagt, einen Einfluss auf die Diversität der Lösungen. Bei Verwendung eines $q_0 = 0.9$ wie beim Ant Colony System, zeigen die nicht-dominierten gültigen Lösungen zwar immer noch eine stärkere Diversität als in den Läufen mit Genetischen Algorithmen, sie liegen jedoch deutlich dichter zusammen als bei Läufen mit $q_0 = 0.4$ (vgl. Abbildung 7.20).

7.3.3. Zusammenfassung

Beim Test der Algorithmen für das mehrkriterielle Problem ergab sich die Schwierigkeit, dass das gewählte Testszenario so ungünstig strukturiert ist, dass die Umgebung des Optimums für eines der drei Zielkriterien sehr viele ungültige Lösungen enthält. Alle der untersuchten Algorithmen scheiterten daran, diese Region zu erreichen, wenn nicht schon in der Startpopulation bzw. der Lösungsmenge der ersten Iteration gültige Lösungen in der Nähe dieses Optimums vorhanden waren.

Davon abgesehen zeigten sich im mehrkriteriellen Fall die Ameisenalgorithmen den Genetischen Algorithmen durch eine höhere Diversität überlegen. Die Lösungen, die in den Testläufen mit Genetischen Algorithmen gefunden wurden, liegen mit beiden untersuchten Algorithmen nahe beieinander. Da ungültige Bereiche des Lösungsraumes nur schwer zu durchqueren sind, konzentriert sich die Suche auf einzelne gültige Inseln. Durch Verkleinerung der Elternmenge beim MOGA-Jas, was zu einer strengeren Auslese führt, lässt sich die Diversität erhöhen. Die

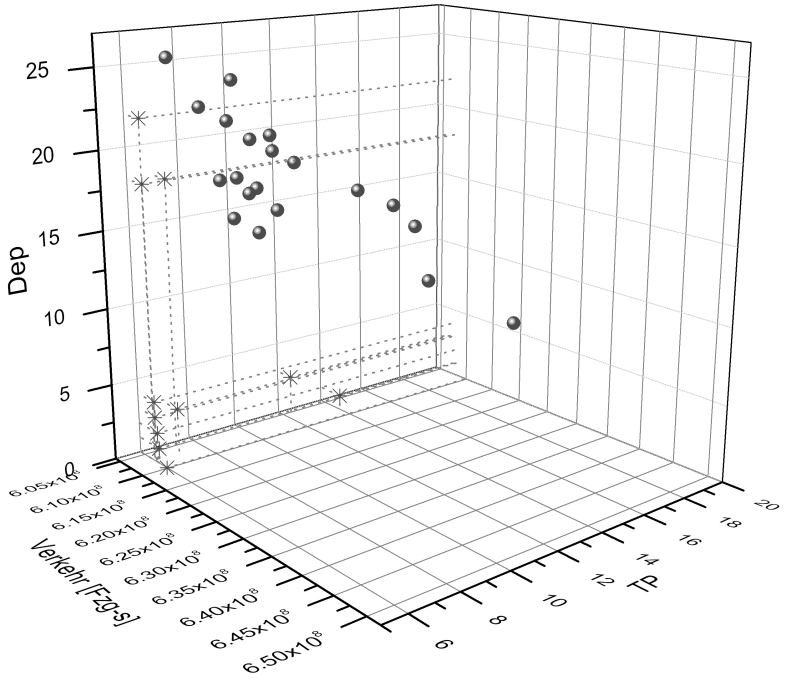

Abbildung 7.19.: Die nicht-dominierten Lösungen nach 200 Iterationen mit PACO mit $m = 10$, $\rho = 0.1$ und $q_0 = 0.4$. Die Lösungen zeigen eine starke Diversität. Von den Schwierigkeiten im Erreichen gute Werte für das Zielkriterium Dep abgesehen (vgl. Abschnitt 7.3), zeigen die Werte zudem eine gute Annäherung an die it einem Stern markierten nicht-dominierten Lösungen aus allen Testläufen.

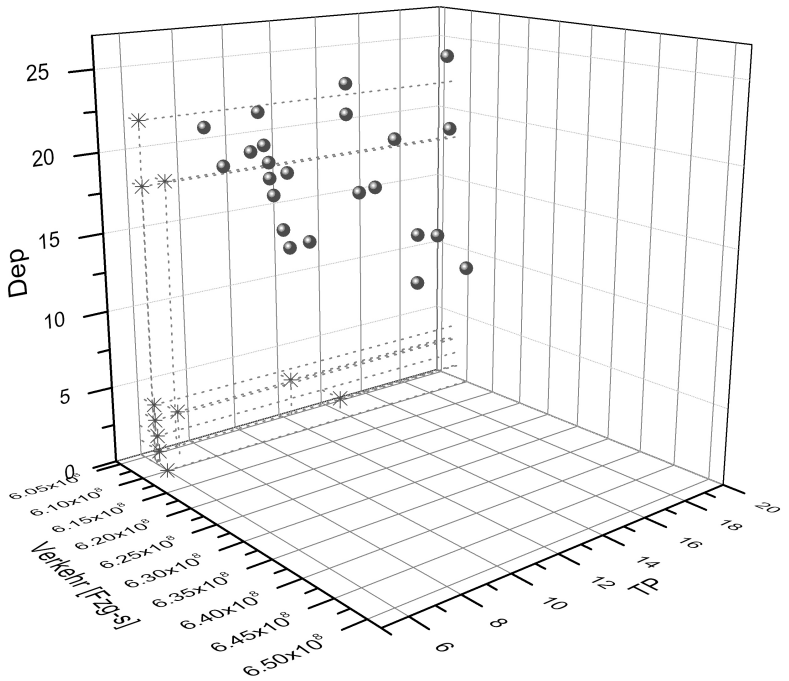

Abbildung 7.20.: Die nicht-dominierten Lösungen nach 200 Iterationen mit PACO mit $m = 10$, $\rho = 0.1$ und $q_0 = 0.9$. Die Lösungen zeigen zwar immer noch eine große Diversität, liegen jedoch deutlich dichter geballt als bei Läufen mit $q_0 = 0.4$ (Abbildung 7.19).

Anzahl der gefundenen nicht-dominierten Lösungen ist jedoch beim MOGA-Jas sehr klein.

Das Bi-Criterion Ant System hat ebenfalls Schwierigkeiten mit der Struktur des Lösungsraumes mit den vielen ungültigen Bereichen. Dies verlangsamt die Suche. Da beim Bi-Criterion Ant System pro Iteration eine große Anzahl an Zielfunktionsauswertungen durchgeführt werden müssen, können in der Praxis nur wenige Iterationen durchgeführt werden. In diesen wenigen Iterationen ist es aufgrund des langsamen Fortschritts der Suche jedoch nicht möglich, gute Lösungen zu erreichen.

Von allen untersuchten Algorithmen zur mehrkriteriellen Suche zeigte PACO die besten Ergebnisse: Durch die kleine Anzahl an benötigten Ameisen kann gegenüber dem Bi-Criterion Ant System in gleicher Zeit eine größere Anzahl an Iterationen durchgeführt werden. Zudem sorgt das strenge Auswahlkriterium dafür, dass kaum ungültige Lösungen auftreten. Die in jeder Iteration neu zufällig bestimmten Wichtungsfaktoren bewirken dagegen, dass trotz strenger Auswahlregeln eine hohe Diversität herrscht. Allerdings konnte auch PACO nicht mit der Struktur des Lösungsraumes in der Nähe des Optimums des dritten Zielkriteriums umgehen.

Zusammenfassend lässt sich feststellen, dass bisher keine wirksamen Ansätze zur Behandlung mehrkriterieller Probleme mit Randbedingungen existieren. Alle untersuchten Algorithmen hatten Schwierigkeiten damit, die Suche nicht zu weit in ungültige Bereiche zu führen, andererseits ungültige Regionen zu durchqueren, um andere gültige Bereiche zu erreichen.

8. Zusammenfassung und Ausblick

Ziel dieser Arbeit war die Entwicklung von Methoden, die Betreiber von Infrastrukturbauwerken bei der Erstellung optimaler Instandsetzungsstrategien unterstützen. Dabei wurde ein Schwerpunkt auf Bauwerke in engmaschigen Straßennetzen, wie sie vornehmlich in städtischen Umgebungen vorkommen, gelegt. Anders als bei Bauwerken an Überlandstraßen muss hier verstärkt auch auf den Straßenverkehr geachtet werden, da ungünstige Kombinationen von Störungen im Netz hier leicht zu einem Verkehrsinfarkt führen können.

Neben dem Einfluss auf den Verkehr wurden noch weitere Zielkriterien identifiziert. Zum einen sollen wichtige Ausfallstraßen nicht mehrfach in kurzen Zeitabständen Störungen durch Instandsetzungsarbeiten unterworfen werden. Bauwerke, die an einer dieser Straßen liegen, sollten also soweit wie möglich zusammengefasst werden. Zudem tritt es gerade in städtischen Netzen häufig auf, dass die Bauwerke noch von dritter Seite, beispielsweise einem Straßenbahnbetreiber, genutzt werden. Auch diese Partei muss Instandsetzungsarbeiten durchführen. Eine Abstimmung der Instandsetzungstermine kann zu massiven Einsparungen auf beiden Seiten führen.

Die Kosten, die in vielen verwandten Arbeiten als, oft einziges, Zielkriterium der Optimierung betrachtet werden, gehen hier als Randbedingungen ein. Damit können die Instandsetzungszeitpläne so gesteuert werden, dass pro Jahr etwa die selbe Summe in Instandsetzungsarbeiten investiert wird, was zu einer höheren Planungssicherheit führt. Als weitere Randbedingung geht in die Optimierung die Bauwerkssicherheit, beschrieben durch Instandsetzungsfristen, ein.

Durch Anforderungen aus der Praxis lässt sich das Problem noch weiter eingrenzen: Eingeschränkte Ressourcen führen dazu, dass pro Jahr nur eine feste Anzahl an Bauwerken instand gesetzt werden kann. Die Beschränkung der Instandsetzungsarbeiten auf die Sommermonate berechtigen zu der Annahme, dass alle für ein Jahr terminierten Arbeiten zur gleichen Zeit durchgeführt werden.

Das so definierte Problem zeigt große Ähnlichkeit mit dem multiplen Rucksackproblem, weist gegenüber diesem allerdings zusätzliche Randbedingungen auf, die eine direkte Übertragung der auf dieses angewendeten Lösungsmethoden unmöglich macht. Wie das Rucksackproblem ist auch das Instandsetzungsproblem *np*-hart. Daher wurden zur seiner Lösung im Rahmen dieser Arbeit metaheuristische Methoden gewählt.

Untersucht wurden sowohl Methoden zur einkriteriellen (mit dem Verkehrseinfluss als einzigem Zielkriterium) als auch zur mehrkriteriellen (mit allen drei identifizierten Zielkriterien) Problemlösung. Als geeignete Metaheuristiken wurden Genetische Algorithmen und Ameisenalgorithmen gewählt. Beide Strategien mussten entsprechend der Problemstruktur adaptiert werden. Dazu wurden geeignete Problemrepräsentationen entwickelt, die den Lösungsraum vollständig abdecken und ihn auf das Problem abgestimmt in möglichst günstiger Weise strukturieren. Das heißt, die Repräsentationen sollen möglichst so beschaffen sein, dass verwandte Lösungen in ihnen benachbart sind und so die Lösungsstrategien sinnvoll angewandt werden können.

Zur Auswertung der Zielfunktion der minimalen Verkehrsstörung wurden die Algorithmen mit einem Verkehrssimulator gekoppelt. Da dessen Aufruf verhältnismäßig rechenzeitintensiv ist, kann die Zielfunktionsauswertung dabei nicht, wie bei metaheuristischen Verfahren sonst üblich, mehrere zehntausend Mal erfolgen, um in der praktischen Anwendung Ergebnisse innerhalb einer akzeptablen Zeitspanne zu erhalten. Daher lag ein besonderes Augenmerk darauf, solche Algorithmen zu finden, die innerhalb weniger Iterationen gute Lösungen erreichen oder aber mit sehr kleinen Lösungspopulationen arbeiten.

Die entwickelten Algorithmen wurden auf verschiedenen Testszenarien untersucht, um ihre grundsätzliche Eignung für derartige Probleme zu zeigen, sowie die einzelnen Algorithmen untereinander zu vergleichen.

Auf dem einkriteriellen Problem zeigten sich dabei die Genetischen Algorithmen den Ameisenalgorithmen überlegen, indem sie näher am globalen Optimum gelegene Lösungen fanden. Allerdings kam auch einer der untersuchten Ameisenalgorithmen, das Ant Colony System, verhältnismäßig nahe an diese Lösungen heran. Zusätzlich zeichnet sich das Ant Colony System durch eine geringe Laufzeit aus, weshalb es hier für eine praktische Anwendung empfohlen wird.

Auf dem mehrkriteriellen Problem wiesen die Genetischen Algorithmen nur eine sehr geringe Diversität in den von ihnen gefundenen Lösungen auf. Hier zeigten sich die Ameisenalgorithmen, namentlich die Pareto Ant Colony Optimization, als überlegen, da die hiermit gefundene Front nicht-dominierter Lösungen eine weite Abdeckung des Lösungsraumes aufweist.

Bei der Bearbeitung des mehrkriteriellen Problems zeigte sich jedoch eine zusätzliche Schwierigkeit: Keiner der untersuchten Algorithmen ist darauf ausgelegt, dass das Optimierungsproblem Randbedingungen aufweist. Die im Rahmen dieser Arbeit entwickelten Strategien zur Behandlung ungültiger Lösungen im mehrkriteriellen Fall erwiesen sich als nicht ausreichend. Ein Teil der Pareto-Front des Testproblems konnte nur in einem geringen Bruchteil der Testläufe erreicht werden und das nur dann, wenn sich bereits die Startlösungen in diesem Bereich des Lösungsraumes befanden.

Da ein Großteil der in der Praxis auftretenden Optimierungsprobleme nicht nur mehrere Zielkriterien sondern auch Randbedingungen aufweist, besteht großer Forschungsbedarf zur Entwicklung geeigneter Strategien zum Umgang mit ungültigen Lösungen. Bisher existieren in der Literatur keine Beispiele für mehrkriterielle Optimierung an Problemen mit stark beschränktem Lösungsraum. Die Erfahrungen in dieser Arbeit zeigen jedoch, dass eine einfache Übertragung der Konzepte zur Randbedingungsbehandlung aus dem einkriteriellen auf den mehrkriteriellen Fall nicht ausrechend ist, da sich die Suche nach der Paretofront grundsätzlich anders verhält als die nach einer einzelnen optimalen Lösung.

Es ist des weiteren zu untersuchen, ob in der Instandsetzungsplanung in der Praxis noch weitere implizit oder explizit formulierte Zielkriterien oder Randbedingungen existieren. Diese müssen nach ihrer Identifizierung mathematisch formuliert und in das Optimierungsproblem überführt werden. Einige der hier beschriebenen Optimierungsalgorithmen müssen dazu weiter angepasst werden, um auch Probleme mit mehr als drei Zielkriterien behandeln zu können.

Für die praktische Anwendung vor allem der mehrkriteriellen Optimierung ist eine bessere Aufbereitung der Ergebnisse wünschenswert. Hier besteht Forschungsbedarf zur Entwicklung geeigneter Benutzeroberflächen, die es dem Nutzer ermöglichen, sich einen guten Überblick über die gefundenen Lösungen zu verschaffen und so die für ihn beste Lösung auszuwählen. Mit in diesen Themenkomplex fällt auch die Entwicklung von Algorithmen zur Vorauswahl der Lösungen, um den Nutzer nicht durch zu viele Informationen zu überfordern. Zu diesem Zweck ist es nötig, die Kriterien, nach denen der Nutzer seine Wunschlösung aussucht, genauer zu untersuchen und so zu identifizieren, welche Lösungen als gleichwertig betrachtet werden.

Die in dieser Arbeit entwickelten Algorithmen sind mit wenigen Änderungen auch auf andere ähnliche Probleme anwendbar. Solche Probleme wären zum Beispiel Arbeiten am Straßenbelag, Sanierungsarbeiten in Gebäuden unter Betrieb aber auch andere Probleme, bei denen Arbeiten mit festen Endfristen auf bestimmte Zeitfenster aufzuteilen sind.

A. Testszenarien

A.1. Szenario 1

Globales Optimum: $1.9089 * 10^8$ Fzgs

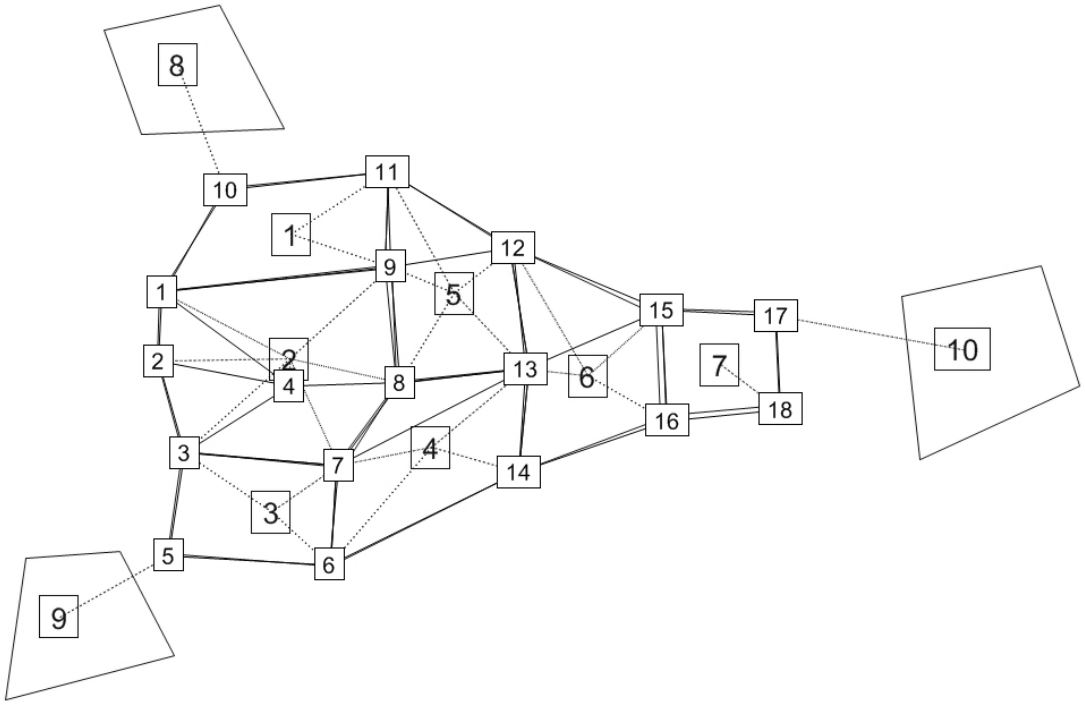

Abbildung A.1.: Netz von Szenario 1

Tabelle A.1.: Bezirke von Szenario 1

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	-	500	400	300	800	1000	800	1000	800	700
2	500	-	900	200	700	800	600	900	600	200
3	400	900	-	900	300	500	700	800	700	800
4	300	200	900	-	500	800	300	400	500	500
5	800	700	300	500	-	900	300	400	900	300
6	1000	800	500	800	900	-	200	800	1000	500
7	800	600	700	300	300	200	-	900	800	900
8	1000	900	800	400	400	800	900	-	300	200
9	800	600	700	500	900	1000	800	300	-	900
10	700	200	800	500	300	500	900	200	900	-

Tabelle A.2.: Straßen von Szenario 1

Nummer	*Anfangsknoten*	*Endknoten*	*Kapazität*	*Bauwerksnummer*
1	1	2	1500	0
2	1	4	200	1
3	2	4	200	2
4	2	3	1500	3
5	3	4	4500	4
6	3	5	4500	5
7	5	6	2000	6
8	6	7	200	7
9	3	7	400	8
10	4	8	4500	9
11	7	8	1000	10
12	6	14	1500	11
13	7	13	1500	12
14	13	14	600	13
15	8	13	4500	14
16	8	9	2500	15
17	1	9	200	16
18	1	10	2500	17
19	10	11	3500	18
20	9	11	200	19
21	11	12	2500	20
22	9	12	3500	21
23	12	13	400	22

Nummer	*Anfangsknoten*	*Endknoten*	*Kapazität*	*Bauwerksnummer*
24	12	15	4000	23
25	13	15	2500	24
26	14	16	2500	25
27	15	16	200	26
28	15	17	4500	27
29	17	18	3500	28
30	16	18	2500	29

Da Szenario 1 nur für das einkriterielle Problem angewendet wurde, sind keine Werte für Instandsetzungsarbeiten von Dritten und Bauwerksgruppen definiert.

Tabelle A.3.: Bauwerke von Szenario 1

Nummer	*Frist*	*Kosten*
0	8	302167
1	6	287060
2	15	112717
3	8	266182
4	10	259884
5	14	153277
6	10	175590
7	2	293344
8	1	163942
9	13	308794
10	3	291267
11	8	399340
12	2	209689
13	4	149632
14	6	379684
15	4	302460
16	9	101667
17	5	281275
18	10	297121
19	9	237382
20	12	251969
21	1	328203
22	5	177978
23	8	343057
24	12	162153
25	6	395536
26	15	347807

Nummer	*Frist*	*Kosten*
27	14	188297
28	3	140642
29	10	355442

A.2. Szenario 2

Beste bekannte Lösung: $6.0526 * 10^8$ Fzgs
Ideale Lösung für das mehrkriterielelle Problem: $6.0526 * 10^8$ Fzgs; TP: 6; Dep: 0

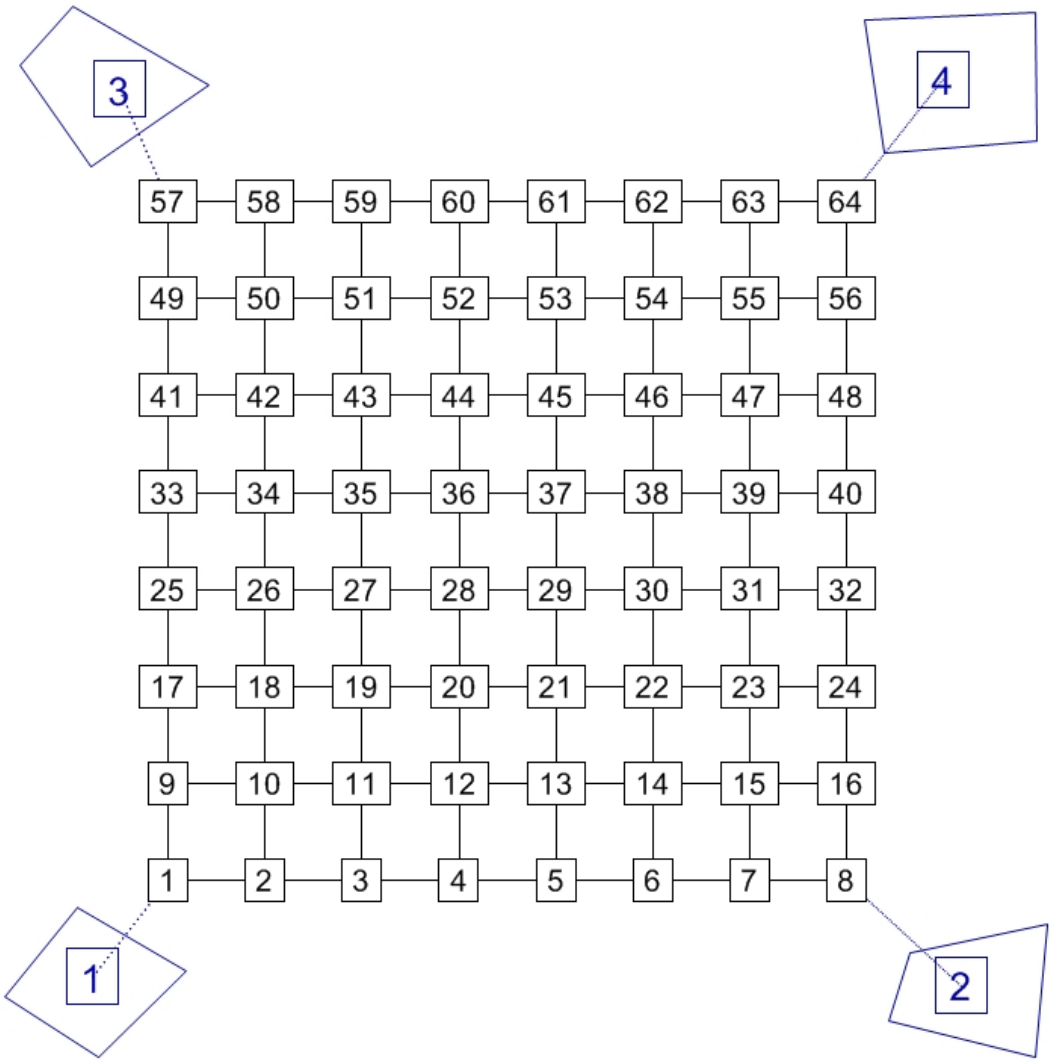

Abbildung A.2.: Netz von Szenario 2

Tabelle A.4.: Bezirke von Szenario 2

	1	**2**	**3**	**4**
1	-	1000	1000	1000
2	1000	-	1000	1000
3	1000	1000	-	1000
4	1000	1000	1000	-

Da beide Fahrtrichtungen der Straßen in diesem Szenario die selbe Kapazität haben, wird in dieser Tabelle nur eine Richtung angegeben.

Tabelle A.5.: Straßen von Szenario 2

Nummer	*Anfangsknoten*	*Endknoten*	*Kapazität*	*Bauwerksnummer*
1	1	2	1000	0
2	2	3	1000	1
3	3	4	1000	2
4	4	5	1000	3
5	5	6	1000	4
6	6	7	1000	5
7	7	8	1000	6
8	9	10	1000	7
9	10	11	1000	8
10	11	12	1000	9
11	12	13	1000	10
12	13	14	1000	11
13	14	15	1000	12
14	15	16	1000	13
15	17	18	1000	14
16	18	19	1000	15
17	19	20	1000	16
18	20	21	1000	17
19	21	22	1000	18
20	22	23	1000	19
21	23	24	1000	20
22	25	26	1000	21
23	26	27	1000	22
24	27	28	1000	23
25	28	29	1000	24
26	29	30	1000	25

Nummer	*Anfangsknoten*	*Endknoten*	*Kapazität*	*Bauwerksnummer*
27	30	31	1000	26
28	31	32	1000	27
29	33	34	1000	28
30	34	35	1000	29
31	35	36	1000	30
32	36	37	1000	31
33	37	38	1000	32
34	38	39	1000	33
35	39	40	1000	34
36	41	42	1000	35
37	42	43	1000	36
38	43	44	1000	37
39	44	45	1000	38
40	45	46	1000	39
41	46	47	1000	40
42	47	48	1000	41
43	49	50	1000	42
44	50	51	1000	43
45	51	52	1000	44
46	52	53	1000	45
47	53	54	1000	46
48	54	55	1000	47
49	55	56	1000	48
50	57	58	1000	49
51	58	59	1000	50
52	59	60	1000	51
53	60	61	1000	52
54	61	62	1000	53
55	62	63	1000	54
56	63	64	1000	55
57	1	9	1000	56
58	9	17	1000	57
59	17	25	1000	58
60	25	33	1000	59
61	33	41	1000	60
62	41	49	1000	61
63	49	57	1000	62
64	2	10	1000	63
65	10	18	1000	64
66	18	26	1000	65
67	26	34	1000	66

Nummer	*Anfangsknoten*	*Endknoten*	*Kapazität*	*Bauwerksnummer*
68	34	42	1000	67
69	42	50	1000	68
70	50	58	1000	69
71	3	11	1000	70
72	11	19	1000	71
73	19	27	1000	72
74	27	35	1000	73
75	35	43	1000	74
76	43	51	1000	75
77	51	59	1000	76
78	4	12	1000	77
79	12	20	1000	78
80	20	28	1000	79
81	28	36	1000	80
82	36	44	1000	81
83	44	52	1000	82
84	52	60	1000	83
85	5	13	1000	84
86	13	21	1000	85
87	21	29	1000	86
88	29	37	1000	87
89	37	45	1000	88
90	45	53	1000	89
91	53	61	1000	90
92	6	14	1000	91
93	14	22	1000	92
94	22	30	1000	93
95	30	38	1000	94
96	38	46	1000	95
97	46	54	1000	96
98	54	62	1000	97
99	7	15	1000	98
100	15	23	1000	99
101	23	31	1000	100
102	31	39	1000	101
103	39	47	1000	102
104	47	55	1000	103
105	55	63	1000	104
106	8	16	1000	105
107	16	24	1000	106
108	24	32	1000	107

Nummer	*Anfangsknoten*	*Endknoten*	*Kapazität*	*Bauwerksnummer*
109	32	40	1000	108
110	40	48	1000	109
111	48	56	1000	110
112	56	64	1000	111

Tabelle A.6.: Bauwerke von Szenario 2

Nummer	*Frist*	*Kosten*	*Arbeiten durch Dritte*	*Gruppe*
0	15	276813	13	-
1	10	283571	14	-
2	8	175413	11	1
3	14	323099	7	-
4	7	245794	-	-
5	9	186502	4	1
6	15	100724	-	-
7	13	374924	-	-
8	8	153449	13	-
9	15	262349	-	1
10	14	360071	-	-
11	10	260778	-	-
12	10	271441	-	-
13	13	346152	10	-
14	10	143094	5	-
15	9	271226	8	-
16	9	356477	-	-
17	8	294633	10	-
18	15	146758	11	-
19	14	336440	12	1
20	8	229205	-	-
21	8	255379	-	-
22	10	121272	1	-
23	9	375695	-	-
24	5	334657	15	-
25	11	263325	3	-
26	12	173999	7	4
27	9	127274	-	-
28	6	139972	9	2
29	12	238222	10	-
30	14	348431	14	2
31	5	178150	6	-
32	10	239770	-	-

Nummer	*Frist*	*Kosten*	*Arbeiten durch Dritte*	*Gruppe*
33	11	105464	4	-
34	12	112406	14	-
35	13	221686	-	-
36	9	282776	12	-
37	14	165837	12	-
38	7	382575	10	-
39	6	143024	13	-
40	11	369528	-	2
41	9	206618	6	-
42	14	398524	-	4
43	14	305387	-	2
44	8	144784	-	-
45	7	292913	-	-
46	6	306319	-	-
47	9	149803	-	-
48	10	361385	2	-
49	7	135615	-	-
50	14	315876	-	-
51	8	313703	-	5
52	11	149668	12	-
53	8	358998	-	-
54	10	393884	-	-
55	15	292245	-	-
56	3	186596	-	4
57	2	288728	11	-
58	1	212598	13	5
59	1	254126	-	1
60	1	395741	3	-
61	2	132185	5	-
62	3	319269	8	-
63	14	225133	-	-
64	5	114376	7	-
65	4	181367	8	-
66	2	152820	6	-
67	4	110824	-	6
68	5	218253	-	-
69	7	285985	-	-
70	8	134330	15	-
71	12	384587	-	3
72	6	108377	-	-
73	3	205022	3	4

Nummer	*Frist*	*Kosten*	*Arbeiten durch Dritte*	*Gruppe*
74	6	326313	6	-
75	13	108887	5	-
76	7	399287	-	-
77	6	198450	-	3
78	13	362073	-	-
79	14	352658	4	6
80	4	225730	-	-
81	9	205443	-	-
82	14	186410	5	5
83	10	399980	6	5
84	8	188747	-	1
85	13	114901	9	4
86	11	155725	-	-
87	4	294995	-	-
88	8	332759	14	-
89	14	397694	-	-
90	13	176272	-	-
91	7	329473	-	-
92	12	365584	10	-
93	9	140986	14	-
94	3	176630	-	-
95	9	101287	-	5
96	14	397379	-	-
97	9	352413	-	4
98	8	116845	14	-
99	5	202217	13	-
100	4	132882	-	-
101	2	202948	-	-
102	4	121381	-	-
103	5	119850	-	-
104	12	319080	-	-
105	3	187493	-	-
106	2	153866	-	-
107	1	180455	-	-
108	1	235452	-	-
109	1	219692	-	-
110	2	100301	-	-
111	3	100808	-	-

A.3. Szenario 3

Beste bekannte Lösung: $5.8973 * 10^8$ Fzgs

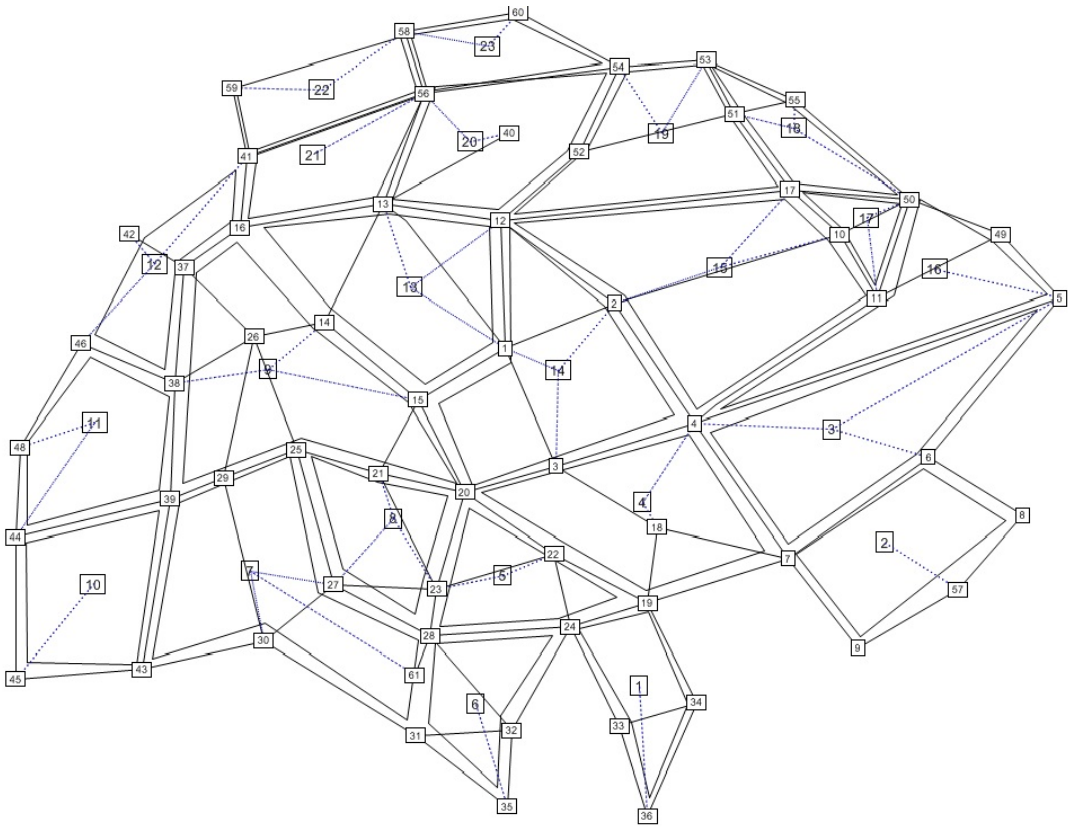

Abbildung A.3.: Netz von Szenario 3

Tabelle A.7.: Bezirke von Szenario 3

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
1	-	25	22	24	25	22	23	20	25	23	23	20	27	26	27	25	10	19	22	23	25	19	22
2	20	-	28	30	27	24	26	24	28	18	26	23	31	29	34	25	13	19	29	25	25	20	24
3	25	28	-	32	26	26	28	27	32	23	26	26	32	30	37	28	20	25	30	24	23	21	26
4	25	27	29	-	27	25	28	26	35	22	25	26	34	31	35	27	20	24	32	29	25	24	28
5	25	25	30	30	-	25	27	25	28	22	24	25	30	28	35	26	17	22	29	26	24	25	30
6	25	24	25	25	25	-	30	26	26	19	30	26	27	25	29	23	12	18	24	27	31	26	29
7	27	28	23	30	30	30	-	26	27	23	25	27	31	30	20	29	15	26	30	32	27	30	27
8	19	24	29	28	25	24	30	-	27	17	23	24	29	27	33	24	13	17	24	25	20	29	25
9	30	30	37	30	30	29	29	27	-	25	28	28	35	27	35	31	20	29	31	29	20	28	23
10	19	24	24	27	26	23	25	23	27	-	22	23	22	23	25	22	11	16	21	22	21	27	21
11	23	23	29	30	25	25	25	20	27	23	-	24	30	24	33	22	12	22	29	27	22	26	19
12	19	24	23	31	27	24	28	23	32	18	24	-	28	26	34	26	12	19	26	24	22	25	23
13	24	23	30	29	29	25	27	26	34	24	28	27	-	34	37	29	20	24	34	29	25	21	29
14	26	28	25	34	31	31	28	25	37	27	28	27	33	-	32	24	16	24	33	29	24	20	25
15	30	31	32	28	31	25	30	29	29	29	29	30	35	34	-	32	23	30	37	33	29	19	23
16	20	24	28	27	24	25	26	26	30	22	23	26	28	30	32	-	14	19	30	26	21	30	27
17	19	19	19	15	17	17	20	19	23	20	15	13	18	18	13	10	-	12	12	13	12	19	18
18	20	21	28	24	23	18	22	16	26	17	20	19	22	24	23	21	14	-	22	22	23	24	18
19	24	27	30	29	28	26	31	27	32	27	27	28	17	36	33	30	22	22	-	26	23	21	25
20	20	26	31	25	26	26	27	25	29	22	25	25	30	32	33	27	20	21	28	-	24	23	27
21	18	25	29	25	24	26	25	20	26	22	23	20	22	26	27	22	17	19	24	28	-	23	26
22	19	20	21	24	25	26	30	29	28	27	26	25	21	20	19	30	19	24	21	23	23	-	25
23	22	24	26	28	30	29	27	25	23	21	19	23	29	25	23	27	18	18	25	27	26	25	-

Tabelle A.8.: Straßen von Szenario 3

Nummer	*Anfangsknoten*	*Endknoten*	*Kapazität Hin*	*Kapazität Rück*	*Bauwerks-nummer*
1	4	5	150	50	0
2	5	6	350	250	1
3	6	7	200	150	2
4	4	7	450	200	3
5	6	8	200	150	4
6	7	9	500	500	5
7	8	57	200	150	6
8	2	4	600	550	7
9	1	2	450	400	8
10	1	3	150	150	9
11	3	4	650	450	10
12	13	14	550	700	11
13	1	12	500	450	12
14	1	13	450	600	13
15	1	15	1500	1350	14
16	9	57	500	500	15
17	12	13	250	350	16
18	13	16	50	100	17
19	38	46	400	400	18
20	2	10	100	250	19
21	10	11	50	100	20
22	4	11	250	250	21
23	2	12	600	350	22
24	12	17	0	0	23
25	10	17	0	50	24
26	3	18	400	600	25
27	18	19	150	150	26
28	7	19	400	300	27
29	7	18	250	50	28
30	15	21	350	300	29
31	15	20	1200	1500	30
32	20	21	250	300	31
33	20	23	1200	1500	32
34	22	23	50	100	33
35	20	22	250	300	34
36	3	20	900	1000	35
37	19	22	100	150	36
38	21	23	100	0	37

Nummer	*Anfangsknoten*	*Endknoten*	*Kapazität Hin*	*Kapazität Rück*	*Bauwerks-nummer*
39	14	26	650	500	38
40	25	26	50	200	39
41	25	27	50	0	40
42	27	28	50	100	41
43	24	28	100	0	42
44	21	25	300	400	43
45	23	27	650	850	44
46	23	28	350	400	45
47	22	24	400	400	46
48	19	24	200	150	47
49	24	32	200	200	48
50	24	33	400	450	49
51	33	34	150	250	50
52	19	34	250	150	51
53	33	36	600	600	52
54	34	36	0	0	53
55	32	35	650	700	54
56	31	35	50	0	55
57	31	32	50	100	56
58	28	32	400	400	57
59	28	61	0	0	58
60	30	31	100	100	59
61	27	30	350	350	60
62	25	29	250	250	61
63	13	56	750	900	62
64	26	29	250	200	63
65	29	30	250	150	64
66	26	38	300	250	65
67	29	39	400	450	66
68	38	39	50	0	67
69	37	38	50	50	68
70	26	37	350	400	69
71	16	37	300	150	70
72	13	40	450	450	71
73	39	44	250	450	72
74	39	43	200	0	73
75	43	45	550	400	74
76	44	45	100	250	75
77	30	43	350	400	76
78	14	15	200	500	77

Nummer	*Anfangsknoten*	*Endknoten*	*Kapazität Hin*	*Kapazität Rück*	*Bauwerks-nummer*
79	50	55	500	450	78
80	37	42	350	300	79
81	16	41	150	300	80
82	51	55	50	0	81
83	53	55	550	550	82
84	51	53	200	150	83
85	53	54	550	550	84
86	52	54	700	500	85
87	46	48	450	400	86
88	17	51	200	250	87
89	44	48	100	50	88
90	5	49	550	550	89
91	11	49	0	0	90
92	49	50	550	550	91
93	17	50	50	50	92
94	10	50	50	150	93
95	11	50	50	50	94
96	12	52	1000	700	95
97	51	52	200	300	96
98	54	56	400	400	97
99	54	60	600	400	98
100	58	60	150	150	99
101	56	58	500	650	-
102	41	56	200	200	-
103	41	59	150	250	-

Tabelle A.9.: Bauwerke von Szenario 3

Nummer	*Frist*	*Kosten*	*Arbeiten durch Dritte*	*Gruppe*
0	14	276813	13	-
1	8	283571	14	-
2	2	175413	11	1
3	9	323099	7	-
4	10	245794	-	-
5	4	186502	4	1
6	1	100724	-	-
7	3	374924	-	-
8	3	153449	13	-
9	13	262349	-	1
10	9	360071	-	-
11	2	260778	-	-

Nummer	*Frist*	*Kosten*	*Arbeiten durch Dritte*	*Gruppe*
12	9	271441	-	-
13	9	346152	10	-
14	4	143094	5	-
15	4	271226	8	-
16	5	356477	-	-
17	10	294633	10	-
18	11	146758	11	-
19	12	336440	12	1
20	3	229205	-	-
21	12	255379	-	-
22	12	121272	1	-
23	14	375695	-	-
24	7	334657	15	-
25	12	263325	3	-
26	11	173999	7	4
27	6	127274	-	-
28	13	139972	9	2
29	7	238222	10	-
30	8	348431	14	2
31	12	178150	6	-
32	3	239770	-	-
33	12	105464	4	-
34	6	112406	14	-
35	9	221686	-	-
36	8	282776	12	-
37	1	165837	12	-
38	8	382575	10	-
39	8	143024	13	-
40	13	369528	-	2
41	9	206618	6	-
42	5	398524	-	4
43	1	305387	-	2
44	4	144784	-	-
45	8	292913	-	-
46	2	306319	-	-
47	7	149803	-	-
48	13	361385	2	-
49	13	135615	-	-
50	13	315876	-	-
51	12	313703	-	5
52	2	149668	12	-

Nummer	*Frist*	*Kosten*	*Arbeiten durch Dritte*	*Gruppe*
53	3	358998	-	-
54	12	393884	-	-
55	5	292245	-	-
56	6	186596	-	4
57	1	288728	11	-
58	10	212598	13	5
59	6	254126	-	1
60	1	395741	3	-
61	12	132185	5	-
62	11	319269	8	-
63	13	225133	-	-
64	11	114376	7	-
65	2	181367	8	-
66	12	152820	6	-
67	3	110824	-	6
68	14	218253	-	-
69	14	285985	-	-
70	4	134330	15	-
71	10	384587	-	3
72	8	108377	-	-
73	13	205022	3	4
74	12	326313	6	-
75	4	108887	5	-
76	4	399287	-	-
77	13	198450	-	3
78	14	362073	-	-
79	4	352658	4	6
80	2	225730	-	-
81	14	205443	-	-
82	6	186410	5	5
83	2	399980	6	5
84	5	188747	-	1
85	15	114901	9	4
86	14	155725	-	-
87	7	294995	-	-
88	1	332759	14	-
89	6	397694	-	-
90	1	176272	-	-
91	5	329473	-	-
92	14	365584	10	-
93	12	140986	14	-

Nummer	*Frist*	*Kosten*	*Arbeiten durch Dritte*	*Gruppe*
94	4	176630	-	-
95	4	101287	-	5
96	12	397379	-	-
97	11	352413	-	4
98	7	116845	14	-
99	11	202217	13	-

A.4. Szenario 4

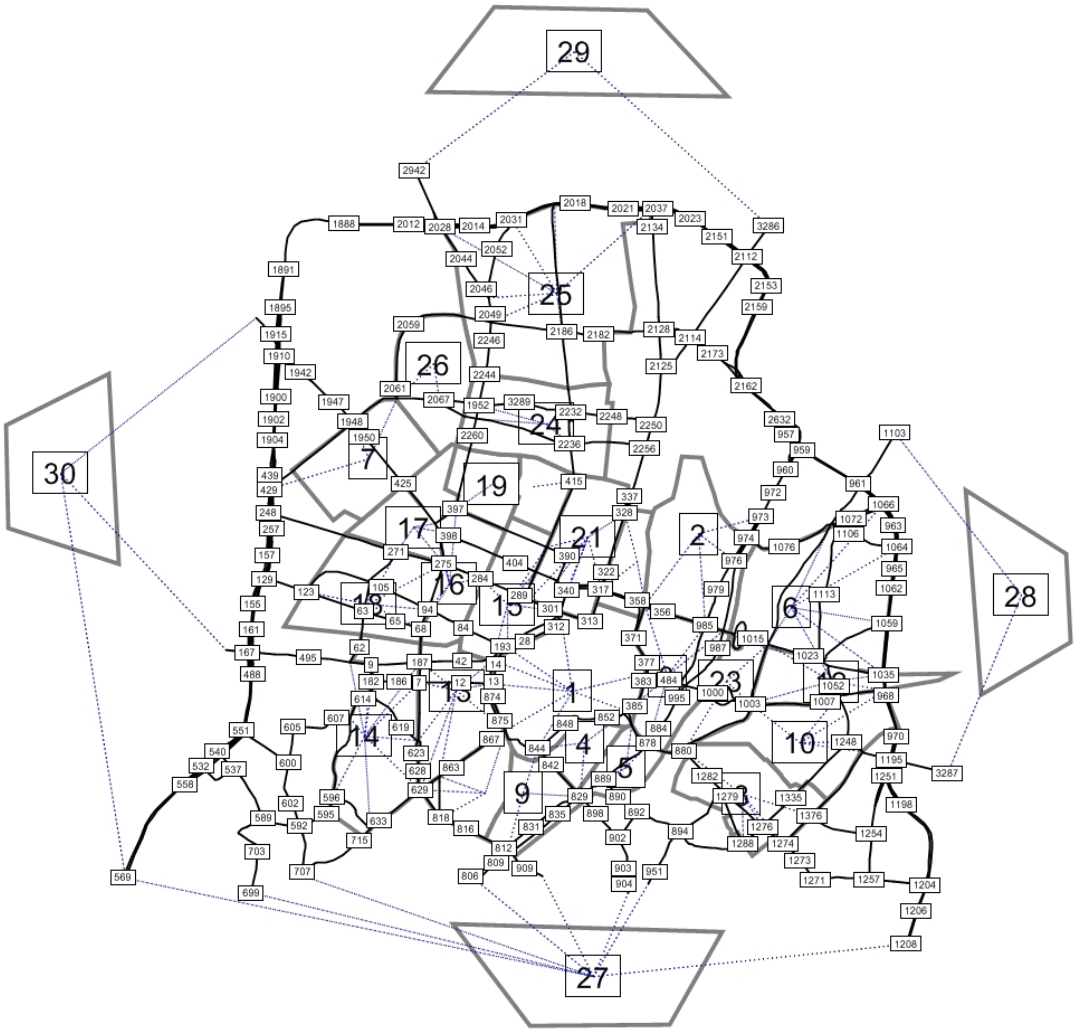

Abbildung A.4.: Netz von Szenario 4

Literaturverzeichnis

Abram, M. (2003). Testbericht SIB-Bauwerke. Forschungsbericht, Kommunales Rechenzentrum Niederrhein.

Akbari, M., Zandieh, M. & Dorri, B. (2012). Scheduling part-time and mixed-skilled workers to maximize employee satisfaction. *The International Journal of Advanced Manufacturing Technology*, S. 1–11.

Angus, D. & Woodward, C. (2009). Multiple objective ant colony optimisation. *Swarm intelligence* 3(1), S. 69–85.

Baker, J. (1987). Reducing bias and inefficiency in the selection algorithm. In: *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, S. 14–21. L. Erlbaum Associates Inc.

Balmer, M., Meister, K., Rieser, M., Nagel, K. & Axhausen, K. W. (2008). Agent-based simulation of travel demand: Structure and computational performance of MATSim-T. In: *2nd TRB Conference on Innovations in Travel Modeling*.

Barán, B. & Schaerer, M. (2003). A multiobjective ant colony system for vehicle routing problem with time windows. In: *21st IASTED International Multi-Conference on Applied Informatics*, S. 97–102.

Bartlett, G. (1995). Genie: A first GA. In: *Practical Handbook of Genetic Algorithms: Applications*, Volume 1, S. 31–56. CRC Press.

Bäck, T. (1995). Generalized Convergence Models for Tournament- and (μ, λ)-Selection. In: *Proceedings of the 6th International Conference on Genetic Algorithms on Genetic algorithms*, S. 2–8. Citeseer.

Bean, J. & ben Hadj-Alouane, A. (1992). A dual genetic algorithm for bounded integer programs. Forschungsbericht 53, Department Industrial and Operations Engeneering, University of Michigan.

Beckmann, M., McGuire, C. & Winsten, C. (1956). *Studies in the Economics of Transportation*. Yale University Press.

Bell, M. & Iida, Y. (1997). *Transportation network analysis*. John Wiley & Sons.

Bellei, G., Gentile, G. & Papola, N. (2005). A within-day dynamic traffic assignment model for urban road networks. *Transportation Research Part B: Methodological* 39(1), S. 1–29.

Birattari, M., Di Caro, G. & Dorigo, M. (2002). Toward the formal foundation of ant programming. In: *Ant algorithms - Proceedings of ANTS 2002 - Third international workshop*, S. 39–72. Springer.

Blickle, T. & Thiele, L. (1996). A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation* 4(4), S. 361–394.

Bocchini, P. & Frangopol, D. (2011). A probabilistic computational framework for bridge network optimal maintenance scheduling. *Reliability Engineering & System Safety* 96(2), S. 332–349.

Bode, M. (2011). Fitness Landscape Analyse zur Entwicklung effektiver und effizienter evolutionärer Algorithmen. In: *23. Forum Bauinformatik, Cork, Irland*.

Bonabeau, E., Theraulaz, G., Deneubourg, J., Aron, S. & Camazine, S. (1997). Self-organization in social insects. *Trends in Ecology & Evolution* 12(5), S. 188–193.

Borrmann, A., Lukas, K., Zintel, M., Schießl, P. & Kluth, M. (2012). BIM-Based Life-Cycle Management for Reinforced Concrete Buildings. *International Journal of 3-D Information Modeling (IJ3DIM)* 1(1), S. 1–24.

Boysen, N. (2010). Ameisenalgorithmen. `http://www.ameisenalgorithmus.de/downloads/ameisenalgorithmen.pdf`. Online; heruntergeladen 19.05.2010.

Braess, D. (1968). Über ein Paradoxon aus der Verkehrsplanung. *Mathematical Methods of Operations Research* 12(1), S. 258–268.

Braun, J. & Wermuth, M. (1973). *VPS 3: Konzept und Programmsystem eines analytischen Gesamtverkehrsmodells*. Institut für Verkehrsplanung und Verkehrswesen der Technischen Universität München.

Brückenweb (2012). `http://www.brueckenweb.de/2content/datenbank/katastrophen/katastrophen.php`. zuletzt besucht 11.09.2012.

Breitenberger, M. (2008). Export von geometrischen Modellen aus AutoCAD. Bachelorarbeit, TU München.

Budai-Balke, G., Dekker, R. & Kaymak, U. (2009). Genetic and memetic algorithms for scheduling railway maintenance activities. Forschungsbericht 30, Erasmus School of Economics (ESE).

Bullnheimer, B., Hartl, R. & Strauss, C. (1997a). A new rank based version of the Ant System. A computational study. Forschungsbericht, SFB Adaptive Information Systems and Modelling in Economics and Management Science, WU Vienna University of Economics and Business.

Bullnheimer, B., Hartl, R. F. & Strauss, C. (1997b). Applying the Ant System to the Vehicle Routing Problem. In: *Proceedings of the 2nd International Conference on Metaheuristics*.

Bullnheimer, B., Hartl, R. F. & Strauss, C. (1999). An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research* 89, S. 319–328.

Burke, E. & Smith, A. (1997). A memetic algorithm for the maintenance scheduling problem. In: *Proceedings of the ICONIP/ANZIIS/ANNES 97 Conference*, S. 469–472. Citeseer.

Burke, E. & Smith, A. (2000). Hybrid evolutionary techniques for the maintenance scheduling problem. *IEEE Transactions on Power Systems* 15(1), S. 122–128.

Černỳ, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications* 45(1), S. 41–51.

Chan, W., Fwa, T. & Zahidul Hoque, K. (2001). Constraint handling methods in pavement maintenance programming. *Transportation Research Part C: Emerging Technologies* 9(3), S. 175–190.

Coello Coello, C. (1999). A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information systems* 1(3), S. 129–156.

Coello Coello, C. (2000a). An updated survey of GA-based multiobjective optimization techniques. *ACM Computing Surveys (CSUR)* 32(2), S. 109–143.

Coello Coello, C. (2000b). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry* 41(2), S. 113–127.

Coello Coello, C. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer methods in applied mechanics and engineering* 191(11-12), S. 1245–1287.

Coello Coello, C., Lamont, G. & Van Veldhuizen, D. (2007). *Evolutionary algorithms for solving multi-objective problems*. Springer-Verlag New York Inc.

Coit, D. & Smith, A. (1996). Penalty guided genetic search for reliability design optimization. *Computers & Industrial Engineering* 30(4), S. 895–904.

Coit, D., Smith, A. & Tate, D. (1996). Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS Journal on Computing* 8, S. 173–182.

Conor, R. (1995). Niche and species formation in Genetic Algorithms. In: *Practical Handbook of Genetic Algorithms: Applications*, Volume 1, S. 57–74. CRC Press.

Cook, S. (1971). The complexity of theorem-proving procedures. In: *Proceedings of the third annual ACM symposium on theory of computing*, S. 151–158. ACM.

Cordón García, O., Herrera, F. & Stützle, T. (2002). A review on the ant colony optimization metaheuristic: basis, models and new trends. *Mathware & Soft Computing* 9(3), S. 141–175.

Darwin, C. (1859). *On the origin of species by means of natural selection.* London: J. Murray.

De Jong, K. (1993). Genetic algorithms are NOT function optimizers. *Foundations of genetic algorithms* 2, S. 5–17.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*, Volume 16. Wiley.

Deb, K. (2004). *Optimization for engineering design: Algorithms and examples.* PHI Learning Pvt. Ltd.

Deeter, D. & Smith, A. (1997). Heuristic optimization of network design considering all-terminal reliability. In: *Annual Reliability and Maintainability Symposium*, S. 194–199. IEEE.

Deneubourg, J., Aron, S., Goss, S. & Pasteels, J. (1990). The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior* 3(2), S. 159–168.

DIN1045 (2008). Tragwerke aus Beton, Stahlbeton und Spannbeton.

DIN1076 (1999). Ingenieurbauwerke im Zuge von Straßen und Wegen, Überwachung und Prüfung, Ausgabe 11/1999.

Doerner, K., Gutjahr, W., Hartl, R., Strauss, C. & Stummer, C. (2004). Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals of Operations Research* 131(1), S. 79–99.

Doerner, K., Hartl, R. & Reimann, M. (2001a). Are COMPETants more competent for problem solving? The case of a multiple objective transportation problem. In: *Proceedings of the GECCO'01*, S. 115–141. Morgan Kaufmann.

Doerner, K., Hartl, R. & Reimann, M. (2001b). Cooperative ant colonies for optimizing resource allocation in transportation. *Applications of Evolutionary Computing* 2037, S. 70–79.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms [auf Italinisch].* Dissertation, Dipartimento di Elettronica, Politecnico di Milano, Mailand.

Dorigo, M. & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation* 1(1), S. 53–66.

Dorigo, M., Maniezzo, V. & Colorni, A. (1991). The ant system: An autocatalytic optimizing process. Forschungsbericht 016, Politecnico di Milano.

Dorigo, M., Maniezzo, V. & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B - Cybernetics* 26(1), S. 29–41.

Dorigo, M. & Stützle, T. (2004). *Ant Colony Optimization* (1. Aufl.). MIT Press.

Dridi, L., Parizeau, M., Mailhot, A. & Villeneuve, J. (2008). Using evolutionary optimization techniques for scheduling water pipe renewal considering a short planning horizon. *Computer-Aided Civil and Infrastructure Engineering* 23(8), S. 625–635.

Duffy, L. (2004). Development of Eirspan: Ireland's bridge management system. *Proceedings of the ICE-Bridge Engineering* 157(3), S. 139–146.

Dugge, B. (2006). *Ein simultanes Erzeugungs-, Verteilungs-, Aufteilungs-und Routenwahlmodell*. Dissertation, Technischen Universität Dresden.

Eiben, A. & Ruttkay, Z. (1996). Self-adaptivity for constraint satisfaction: Learning penalty functions. In: *Proceedings of IEEE International Conference on Evolutionary Computation*, S. 258–261. IEEE.

Eiben, A. & Smith, J. (2003). *Introduction to evolutionary computing*. Springer Verlag.

El Moudani, W., Alberto Nunes Cosenza, C., de Coligny, M. & Mora-Camino, F. (2001). A bi-criterion approach for the airlines crew rostering problem. In: *First International Conference on Evolutionary Multi-Criterion Optimization*, S. 486–500. Springer.

Elbehairy, H., Elbeltagi, E., Hegazy, T. & Soudki, K. (2006). Comparison of two evolutionary algorithms for optimization of bridge deck repairs. *Computer-Aided Civil and Infrastructure Engineering* 21(8), S. 561–572.

Eusuff, M., Lansey, K. & Pasha, F. (2006). Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. *Engineering Optimization* 38(2), S. 129–154.

Feo, T. & Resende, M. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters* 8(2), S. 67–71.

Ferreira, A., Antunes, A. & Picado-Santos, L. (2002). Probabilistic segment-linked pavement management optimization model. *Journal of Transportation Engineering* 128(6), S. 568–577.

Fogel, D. (2006). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press.

Fogel, D. & Ghozeil, A. (1997). A note on representations and variation operators. *IEEE Transactions on Evolutionary Computation* 1(2), S. 159–161.

Fogel, L., Owens, A. & Walsh, M. (1966). *Artificial intelligence through simulated evolution*. John Wiley.

Frangopol, D., Kong, J. & Gharaibeh, E. (2001). Reliability-based life-cycle management of highway bridges. *Journal of Computing in Civil Engineering* 15(1), S. 27–34.

Fwa, T. & Farhan, J. (2012). Optimal Multi-Asset Maintenance Budget Allocation in Highway Asset Management. *Journal of Transportation Engineering* 138.

Fwa, T., Tan, C. & Chan, W. (1994). Road-Maintenance Planning Using Genetic Algorithms. II: Analysis. *Journal of transportation engineering* 120(5), S. 710–722.

Gagné, C., Price, W. & Gravel, M. (2001). Scheduling a single machine with sequence dependent setup time using ant colony optimization. In: *Proceedings of the fourth Metaheuristics International Conference MIC01*, S. 79–84.

Gambardella, L. & Dorigo, M. (1996). Solving symmetric and asymmetric TSPs by ant colonies. In: *Proceedings of IEEE International Conference on Evolutionary Computation*, S. 622–627. IEEE.

Gambardella, L., Taillard, É. & Agazzi, G. (1999). MACS-VRTPW: A multiple ant colony system for vehicle routing problems with time windows. In: *New ideas in optimization*, S. 63–76. McGraw-Hill.

Gandomi, A., Yang, X. & Alavi, A. (2011). Mixed variable structural optimization using Firefly Algorithm. *Computers & Structures* 89, S. 2325–2336.

Gandomi, A., Yang, X. & Alavi, A. (2012). Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Engineering with Computers*.

Gao, L., Xie, C., Zhang, Z. & Waller, S. (2012). Network-Level Road Pavement Maintenance and Rehabilitation Scheduling for Optimal Performance Improvement and Budget Utilization. *Computer-Aided Civil and Infrastructure Engineering* 27, S. 276–287.

Garber, N. & Hoel, L. (2010). *Traffic and highway engineering* (4 Aufl.). Cengage Learning.

García-Martínez, C., Cordón, O. & Herrera, F. (2004). An empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. In: *Proceedings of the 4th International Workshop on Ant Colony Optimization and Swarm Intelligence*, S. 61–72. Springer.

Garcia-Martinez, C., Cordón, O. & Herrera, F. (2007). A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. *European Journal of Operational Research* 180(1), S. 116–148.

Garey, M. & Johnson, D. (1979). *Computers and intractability*. Freeman San Francisco, CA.

Gehlen, C. (2000). *Probabilistische Lebensdauerbemessung von Stahlbetonbauwerken - Zuverlässigkeitsbetrachtungen zur wirksamen Vermeidung von Bewehrungskorrosion*. Dissertation, RWTH Aachen.

Gehlen, C., Schießl, P. & Schießl-Pecka, A. (2008). Hintergrundinformationen zum Positionspapier des DAfStb zur Umsetzung des Konzepts von leistungsbezogenen Entwurfsverfahren unter Berücksichtigung von DIN EN 206-1, Anhang. *Beton- und Stahlbetonbau* 103(12), S. 840–851.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research* 13(5), S. 533–549.

Glover, F. (1989). Tabu search-part I. *ORSA Journal on computing* 1(3), S. 190–206.

Glover, F. (1990). Tabu search-part II. *ORSA Journal on computing* 2(1), S. 4–32.

Goldberg, D. & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms* 1, S. 69–93.

Goldberg, D. & Lingle Jr, R. (1985). Alleles Loci and the Traveling Salesman Problem. In: *Proceedings of the 1st International Conference on Genetic Algorithms*, S. 154–159. L. Erlbaum Associates Inc.

Goldberg, D. & Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In: *Proceedings of the 2nd International Conference on Genetic Algorithms*, S. 41–49. L. Erlbaum Associates Inc.

González, J., Romera, R., Carretero Pérez, J. & Pérez, J. (2006). Optimal railway infrastructure maintenance and repair policies to manage risk under uncertainty with adaptive control. Forschungsbericht, Universidad Carlos III de Madrid.

Goss, S., Aron, S., Deneubourg, J. & Pasteels, J. (1989). Self-organized shortcuts in the Argentine ant. *Naturwissenschaften* 76(12), S. 579–581.

Gravel, M., Price, W. & Gagne, C. (2002). Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic. *European Journal of Operational Research* 143(1), S. 218–229.

Graybeal, B., Phares, B., Rolander, D., Moore, M. & Washer, G. (2002). Visual inspection of highway bridges. *Journal of nondestructive evaluation* 21(3), S. 67–83.

Guignier, F. & Madanat, S. (1999). Optimization of infrastructure systems maintenance and improvement policies. *Journal of Infrastructure Systems* 5(4), S. 124–134.

Guntsch, M. & Middendorf, M. (2003). Solving multi-criteria optimization problems with population-based ACO. In: *Proceedings of the second international conference on evolutionary multi-criterion optimization*, S. 464–478. Springer.

Gutjahr, W. (2000). A graph-based ant system and its convergence. *Future Generation Computer Systems* 16(8), S. 873–888.

Haardt, P. (1999). *Algorithmen zur Zustandsbewertung von Ingenieurbauwerken.* Number 22 in Berichte der Bundesanstalt für Straßenwesen, Brücken- und Ingenieurbau. Bundesanstalt für Straßenwesen.

Halfawy, M., Dridi, L. & Baker, S. (2008). Integrated decision support system for optimal renewal planning of sewer networks. *Journal of Computing in Civil Engineering* 22(6), S. 360–372.

Haller, H. & Bascuro, G. (2006). *KUBA-DB Benutzerhandbuch.*

Hamm, M. & König, M. (2010). Constraint-based Multi-objective Optimization of Construction Schedules. In: *Proceedings of the International Conference Computing in Civil and Building Engineering*, Volume 30, S. 243.

Hammad, A., Zhang, C., Hu, Y. & Mozaffari, E. (2006). Mobile Model-Based Bridge Lifecycle Management System. *Computer-Aided Civil and Infrastructure Engineering* 21(7), S. 530–547.

Hawk, H. (1999). Bridgit: user-friendly approach to bridge management. *Transportation Research Circular* 498, S. E–7/1 – E–7/15.

He, J., Liu, Y. Q., Chen, A. & Yoda, T. (2012). Study on performance evaluation and maintenance management system of weathering steel bridge. In: *Proceedings of the 6th International Conference on Bridge Maintenance, Safety and Management (IABMAS)*, S. 1288–1294.

Henriksen, A. (1999). Bridge Management - Routine Maintenance: Recent Experience with the Routine Management Module in the DANBRO Bridge Management System. *Transportation Research Circular* 498, S. I–5/1–I–5/13.

Hertkorn, G. (2004). *Mikroskopische Modellierung von zeitabhängiger Verkehrsnachfrage und von Verkehrsflussmustern.* Dissertation, Universität zu Köln.

Heydecker, B. G. (1986). On the definition of traffic equilibrium. *Transportation Research Part B: Methodological* 20(6), S. 435–440.

Hinterding, R. (1994). Mapping, order-independent genes and the knapsack problem. In: *Proceedings of the First IEEE Conference on Evolutionary Computation*, S. 13–17. IEEE.

Hoffmeister, F. & Sprave, J. (1996). Problem-independent handling of constraints by use of metric penalty functions. In: *Proceedings of the Fifth Annual Conference on Evolutionary Programming*, S. 289–294. Citeseer.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* MIT Press.

Holst, R. (2005). *Entwicklung eines Bauwerks-Management-Systems für das deutsche Fernstraßennetz, Stufe 3.* Wirtschaftsverlag NW, Verlag für neue Wissenschaft.

Homaifar, A., Qi, C. & Lai, S. (1994). Constrained optimization via genetic algorithms. *Simulation* 62(4), S. 242–253.

Horn, J. (1996). Multicriteria decision making and evolutionary computation. In: *Handbook of Evolutionary Computation*, Volume 1, S. F1.9:1–F1.9:15. Citeseer.

Horn, J., Nafpliotis, N. & Goldberg, D. (1993). Multiobjective optimization using the niched pareto genetic algorithm. Forschungsbericht 93005, Illinois Genetic Algorithms Laboratory.

Horn, J., Nafpliotis, N. & Goldberg, D. (1994). A niched Pareto genetic algorithm for multiobjective optimization. In: *Proceedings of the First IEEE Conference on Evolutionary Computation*, S. 82–87. IEEE.

Iredi, S., Merkle, D. & Middendorf, M. (2001). Bi-criterion optimization with multi colony ant algorithms. In: *First International Conference on Evolutionary Multi-Criterion Optimization*, S. 359–372. Springer.

Ishibuchi, H. & Murata, T. (1998). A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 28(3), S. 392–403.

Jaszkiewicz, A. (2002). Genetic local search for multi-objective combinatorial optimization. *European journal of operational research* 137(1), S. 50–71.

Joines, J. & Houck, C. (1994). On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. In: *Proceedings of the First IEEE Conference on Evolutionary Computation*, S. 579–584. IEEE.

Kazarlis, S. & Petridis, V. (1998). Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms. In: *Parallel Problem Solving from Nature*, S. 211–220. Springer.

Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization. In: *IEEE International Conference on Neural Networks*, Volume 4, S. 1942–1948. IEEE.

Kihira, H., Senuma, T., Tanaka, M., Nishioka, K., Fujii, Y. & Sakata, Y. (2005). A corrosion prediction method for weathering steels. *Corrosion science* 47(10), S. 2377–2390.

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics* 34(5), S. 975–986.

Kirkpatrick, S., Gelatt Jr, C. & Vecchi, M. (1983). Optimization by simulated annealing. *Science* 220(4598), S. 671–680.

Klinzmann, C. & Hosser, D. (2005). Probabilistic Building Inspection and Life Assessment–a computer program for reliability based system assessment. In: *Proceedings of 22nd CIBW78 Conference Information Technology in Construction, Dresden.*

Kowalczyk, R. (1997). Constraint consistent genetic algorithms. In: *IEEE International Conference on Evolutionary Computation*, S. 343–348. IEEE.

Kuri Morales, A. & Villegas Quezada, C. (1998). A universal eclectic genetic algorithm for constrained optimization. In: *Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing*, Volume 1, S. 518–522.

Kutter, E. (1972). *Demographische Determinanten städtischen Personenverkehrs.* Dissertation, TU Braunschweig.

Lapa, C., Pereira, C. & de Barros, M. (2006). A model for preventive maintenance planning by genetic algorithms based in cost and reliability. *Reliability Engineering & System Safety* 91(2), S. 233–240.

Lapa, C., Pereira, C. & Mol, A. (2000). Maximization of a nuclear system availability through maintenance scheduling optimization using a genetic algorithm. *Nuclear engineering and design* 196(2), S. 219–231.

Le Riche, R. & Haftka, R. (1995). Improved genetic algorithm for minimum thickness composite laminate design. *Composites Engineering* 5(2), S. 143–161.

Lee, H. (2009). Optimizing schedule for improving the traffic impact of work zone on roads. *Automation in Construction* 18(8), S. 1034–1044.

Lee, H. (2011). Renovation scheduling to minimize user impact of a building that remains in operation. *Automation in Construction* 22, S. 398–405.

Lee, S. (1996). A cell transmission based assignment-simulation model for integrated freeway/surface street systems. Masterarbeit, Ohio State University.

Leguizamón, G. & Coello Coello, C. (2011). Multi-Objective Ant Colony Optimization: A Taxonomy and Review of Approaches. In: *Integration of Swarm Intelligence and Artificial Neural Networks*, S. 67–94. World Scientific.

Li, H., Huang, X. & Feng, Q. (2011). Optimizing expressway maintenance planning by coupling ant algorithm and geography information system transportation in Hubei province, China. In: *IEEE International Geoscience and Remote Sensing Symposium*, S. 2977–2979. IEEE.

Li, S., Ting, C., Lee, C. & Chen, S. (2002). Maintenance scheduling of oil storage tanks using tabu-based genetic algorithm. In: *Proceedings. 14th IEEE International Conference on Tools with Artificial Intelligence*, S. 209–215. IEEE.

Liu, C., Hammad, A. & Itoh, Y. (1997). Multiobjective optimization of bridge deck rehabilitation using a genetic algorithm. *Microcomputers in Civil Engineering* 12, S. 431–443.

Liu, M. & Frangopol, D. (2005). Bridge annual maintenance prioritization under uncertainty by multiobjective combinatorial optimization. *Computer-Aided Civil and Infrastructure Engineering* 20(5), S. 343–353.

Lohse, D., Teichert, H., Dugge, B. & Bachner, G. (1997). Ermittlung von Verkehrsströmen mit n-linearen Gleichungssystemen unter Beachtung von Nebenbedingungen einschließlich Parameterschätzung. Forschungsbericht 5, Institut für Verkehrsplanung und Straßenverkehr der TU Dresden.

Louis, S. & Rawlins, G. (1993). Pareto optimality, GA-easiness and deception. In: *Proceedings of the Fifth International Conference on Genetic Algorithms*, S. 118–123.

Lounis, Z. (2005). Network-level bridge management using a multiobjective optimization decision model. In: *First CSCE speciality conference on infrastructure technologies, management and policy*, S. 121–1–121–9.

Lourenço, H., Martin, O. & Stützle, T. (2003). Iterated local search. In: *Handbook of metaheuristics*, S. 320–353. Springer.

Lukas, K. & Borrmann, A. (2012). Integrated bridge management from 3D-model to network level. In: *Proceedings of the 6th International Conference on Bridge Maintenance, Safety and Management (IABMAS)*, S. 3449–3455.

Maier, P., Kuhlmann, U., Popa, N. & Willms, R. (2012). Optimizing bridge design by improved deterioration models through fatigue tests. In: *Proceedings of the 6th International Conference on Bridge Maintenance, Safety and Management (IABMAS)*, S. 1816–1821.

Maniezzo, V. (1999). Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS Journal on Computing* 11(4), S. 358–369.

Maniezzo, V. & Colorni, A. (1999). The ant system applied to the quadratic assignment problem. *IEEE Transactions on Knowledge and Data Engineering* 11(5), S. 769–778.

Mariano, C. & Morales, E. (1999). A multiple objective ant-q algorithm for the design of water distribution irrigation networks. In: *Proceedings of First International Workshop on Ant Colony Optimization ANTS*. Citeseer.

Martin, O., Otto, S. & Felten, E. (1991). Large-Step Markov Chains for the Traveling Salesman Problem. *Complex Systems* 5(3), S. 299–326.

Menke, F.-W. (2010). Verschiedene persönliche Gespräche.

Merkle, D., Middendorf, M. & Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation* 6(4), S. 333–346.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. & Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics* 21, S. 1087–1092.

Michalewicz, Z. (1995a). Genetic algorithms, numerical optimization, and constraints. In: *Proceedings of the Sixth International Conference on Genetic Algorithms*, S. 151–158. Eshelman, Morgan Kaufmann Publisher.

Michalewicz, Z. (1995b). A survey of constraint handling techniques in evolutionary computation methods. In: *Proceedings of the 4th Annual Conference on Evolutionary Programming*, S. 135–155. The MIT Press, Cambridge, Massachusetts.

Michalewicz, Z. & Attia, N. (1994). Evolutionary optimization of constrained problems. In: *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, S. 98–108. World Scientific Publishing, River Edge, NJ.

Michalewicz, Z. & Fogel, D. (2004). *How to solve it: modern heuristics*. Springer-Verlag New York Inc.

Miettinen, K. (1999). *Nonlinear multiobjective optimization*, Volume 12. Springer.

Milachowski, C. (2008). Konzeption eines Geburtszertifikatis als Element eines Lebensdauermanagementsystems. Diplomarbeit, TU München.

Miyamoto, A., Kawamura, K. & Nakamura, H. (2000). Bridge management system and maintenance optimization for existing bridges. *Computer-Aided Civil and Infrastructure Engineering* 15(1), S. 45–55.

Müller, H. S. & Vogel, M. (2009). Lebensdauerprognose für Betonbrücken - Wo stehen wir heute? In: *19. Dresdner Brückenbausymposium*, S. 261–275.

Mockus, J., Eddy, W., Mockus, A., Mockus, L. & Reklaitis, G. (1996). Bayesian Heuristic Approach to Discrete and Global Optimization: Algorithms, Visualization. *Software, and Applications*.

Morcous, G. & Lounis, Z. (2003). A new approach to programming maintenance activities for concrete bridge decks. In: *Annual Conference of the Canadian Society for Civil Engineering*, S. 1–9.

Morcous, G. & Lounis, Z. (2005). Maintenance optimization of infrastructure networks using genetic algorithms. *Automation in Construction* 14(1), S. 129–142.

Morcous, G. & Lounis, Z. (2006). Integration of stochastic deterioration models with multi-criteria decision theory for maintenance optimization of bridge decks. *Special Issue of Canadian Journal of Civil Engineering in Honor of MS Mirza* 33(6), S. 756–765.

Munõz, A., Martorell, S. & Serradell, V. (1997). Genetic algorithms in optimizing surveillance and maintenance of components. *Reliability Engineering & System Safety* 57(2), S. 107–120.

Murata, T. & Ishibuchi, H. (1995). MOGA: Multi-objective genetic algorithms. In: *IEEE International Conference on Evolutionary Computation*, Volume 1, S. 289–294. IEEE.

Negnevitsky, M. & Kelareva, G. (1999). Application of genetic algorithms for maintenance scheduling in power systems. In: *6th International Conference on Neural Information Processing*, Volume 2, S. 447–452. IEEE.

Neves, L., Frangopol, D. & Cruz, P. (2006). Probabilistic lifetime-oriented multiobjective optimization of bridge maintenance: single maintenance type. *Journal of structural engineering* 132(6), S. 991–1005.

Ng, M., Lin, D. & Waller, S. (2009). Optimal Long-Term Infrastructure Maintenance Planning Accounting for Traffic Dynamics. *Computer-Aided Civil and Infrastructure Engineering* 24(7), S. 459–469.

Ning, X., Lam, K. & Lam, M. (2010). Dynamic construction site layout planning using max-min ant system. *Automation in Construction* 19(1), S. 55–65.

Orcesi, A. & Cremona, C. (2010). A bridge network maintenance framework for Pareto optimization of stakeholders/users costs. *Reliability Engineering & System Safety* 95(11), S. 1230–1243.

Ouyang, Y. & Madanat, S. (2004). Optimal scheduling of rehabilitation activities for multiple pavement facilities: exact and approximate solutions. *Transportation Research Part A: Policy and Practice* 38(5), S. 347–365.

Pareto, V. (1896). *Cours d'economie politique*. F. Rouge.

PTV AG (2009a). *VISUM 11.0 - Fundamentals*.

PTV AG (2009b). *VISUM 11.0 - User Manual*.

PTV AG (2011). *VISSIM 5.40-01 User Manual*.

Radcliffe, N. (1991). Forma analysis and random respectful recombination. In: *Proceedings of the fourth international conference on genetic algorithms*, S. 222–229. San Marco CA: Morgan Kaufmann.

RI-EBW-PRÜF (2007, 11). Richtlinie zur einheitlichen Erfassung, Bewertung und Auswertung von Ergebnissen der Bauwerksprüfungen nach DIN 1076.

Rinke, N. (2010). Strategien zur dynamischen Routensuche und Alternativroutenbestimmung mittels genetischer Algorithmen. In: *22. Forum Bauinformatik*, S. 179–188.

Robert, W., Marshall, A., Shepard, R. & Aldayuz, J. (2003). Pontis bridge management system: state of the practice in implementation and development. In: *Proceedings of the 9th International Bridge Management Conference*, S. 49–60.

Scheinberg, T. & Anastasopoulos, P. (2010). Pavement Preservation Programming: A Multi-year, Multi-constraint Optimization Methodology. In: *Transportation Research Board 89th Annual Meeting*, Number 10-1565.

Schießl, P., Gehlen, C., Zintel, M., Rank, E., Borrmann, A., Lukas, K., Budelmann, H., Empelmann, M., Heumann, G., Starck, T. W. & Keßler, S. (2011). *Verbundforschungsvorhaben Nachhaltig Bauen mit Beton Lebenszyklusmanagementsystem zur Nachhaltigkeitsbeurteilung - Teilprojekt D*. Number 586 in Schriftenreihe des Deutschen Ausschusses für Stahlbeton. Deutscher Ausschuss für Stahlbeton.

Schießl, P. & Mayer, T. (2007). *Schlussberichte zur ersten Phase des DAfStb / BMBF-Verbundforschungsvorhabens "Nachhaltig Bauen mit Beton"*. Number 572 in Schriftenreihe des Deutschen Ausschusses für Stahlbeton. Deutscher Ausschuss für Stahlbeton.

Schmiedel, R. (1983). *Bestimmung verhaltensähnlicher Personenkreise für die Verkehrsplanung*. Dissertation, Universität Karlsruhe.

Schnetgöke, R. (2008). *Zuverlässigkeitsorientierte Systembewertung von Massivbauwerken als Grundlage für die Bauwerksüberwachung*. Dissertation, Universität Braunschweig.

Shum, Y. & Gong, D. (2007). The application of genetic algorithm in the development of preventive maintenance analytic model. *The International Journal of Advanced Manufacturing Technology* 32(1), S. 169–183.

Solimanpur, M., Vrat, P. & Shankar, R. (2005). An ant algorithm for the single row layout problem in flexible manufacturing systems. *Computers & Operations Research* 32(3), S. 583–598.

Spatial (2006). *Dokumentation von 3D ACIS Modeler*.

Straub, D. (2004). *Generic approaches to risk based inspection planning for steel structures*. Dissertation, ETH Zürich.

Surry, P., Radcliffe, N. & Boyd, I. (1995). A multi-objective approach to constrained optimisation of gas supply networks: The COMOGA method. In: *Evolutionary Computing*, S. 166–180. Springer.

Tan, K., Lee, T. & Khor, E. (2002). Evolutionary algorithms for multi-objective optimization: performance assessments and comparisons. *Artificial intelligence review* 17(4), S. 251–290.

Tate, D. & Smith, A. (1995). A genetic approach to the quadratic assignment problem. *Computers & Operations Research* 22(1), S. 73–83.

Teodorović, D. & Dell´Orco, M. (2005). Bee colony optimization–a cooperative learning approach to complex transportation problems. In: *Advanced OR and AI Methods in Transportation*, S. 51–60.

Tsunokawa, K. & Schofer, J. (1994). Trend curve optimal control model for highway pavement maintenance: Case study and evaluation. *Transportation Research Part A: Policy and Practice* 28(2), S. 151–166.

Valenzuela-Rendón, M., Uresti-Charre, E. & Monterrey, I. (1997). A non-generational genetic algorithm for multiobjective optimization. In: *Proceedings of the Seventh International Conference on Genetic Algorithms*, S. 658–665. Citeseer.

Vanier, D., Newton, L. & Rahman, S. (2009). *Implementation Details to Support a Generalized Framework for Municipal Infrastructure Management*.

Vesikari, E. (2008). Life-Cycle Management Tools for Civil Infrastructure. In: *Proceedings of the 3rd International Workshop on Lifetime Engineering of Civil Infrastructure*.

Wang, W. & Goldschmidt, P. (2008). Optimisation of Pavement Maintenance and Renewal Activities. In: *30th Conference of the Australian Institutes of Transport Research*.

Wardrop, J. (1952). Some Theoretical Aspects of Road Traffic Research. *Proceedings of the Institution of Civil Engineers* 1, S. 325–378.

Weise, T. (2009). *Global Optimization Algorithms - Theory and Application* (2. Aufl.). Self-Published.

Wermuth, M. (2005). Modellvorstellungen zur Prognose. In: G. Steierwald, H. Künne, & W. Vogt (Hrsg.), *Stadtverkehrsplanung: Grundlagen, Methoden, Ziele* (2 Aufl.)., S. 243–295. Springer.

Wilson, A. (1967). A statistical theory of spatial distribution models. *Transportation Research* 1, S. 253–269.

Wolpert, D. & Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1(1), S. 67–82.

Xiao, J., Michalewicz, Z. & Zhang, L. (1996). Evolutionary planner/navigator: Operator performance and self-tuning. In: *Proceedings of IEEE International Conference on Evolutionary Computation*, S. 366–371. IEEE.

Xiao, J., Michalewicz, Z., Zhang, L. & Trojanowski, K. (1997). Adaptive evolutionary planner/navigator for mobile robots. *IEEE Transactions on Evolutionary Computation* 1(1), S. 18–28.

Yang, X. (2009). Firefly algorithms for multimodal optimization. In: *Stochastic algorithms: foundations and applications*, S. 169–178. Springer.

Yang, X. (2010). A new metaheuristic bat-inspired algorithm. *Nature Inspired Cooperative Strategies for Optimization* 284, S. 65–74.

Zhang, T., Andrews, J. & Wang, R. (2012). Optimal Scheduling of Track Maintenance on a Railway Network. *Quality and Reliability Engineering International*.

Zilch, K., Straub, D., Dier, F. & Fischer, J. (2011). *Entwicklung von Verfahren einer zuverlässigkeitsbasierten Bauwerksprüfung*. Number B 85 in Berichte der Bundesanstalt für Straßenwesen. Bundesanstalt für Straßenwesen.