Hochschule
Bielefeld
University of
Applied Sciences
and Arts

DETERMINISTISCHE UND BIONISCHE HEURISTIKEN ZUR LÖSUNG VON ENTSORGUNGSPROBLEMEN

Am Beispiel des Laubharkproblems

Karim Abdelhak, Bernhard Bachmann, Ralf Derdau,
Andreas Hartmann, Hermann-Josef Kruse

2023

Fachbereich
Ingenieurwissenschaften
und Mathematik

Angewandte Mathematik

Impressum

Forschungsreihe des Fachbereichs
Ingenieurwissenschaften und Mathematik

Band 5

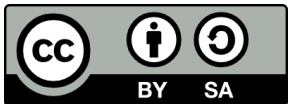
ISSN: 2196-6192
DOI: 10.57720/3321

Verantwortliche und ausführende Stelle (Herausgeber)
im Sinne des §55 I RStV und des Presserechts NRW:
Hochschule Bielefeld (HSBI)
Fachbereich Ingenieurwissenschaften und Mathematik

Dekan Prof. Dr.-Ing. Rolf Naumann

Interaktion 1
33619 Bielefeld

Verantwortliche Redaktion
Prof. Dr. Bernhard Bachmann
Hochschule Bielefeld (HSBI)
Fachbereich Ingenieurwissenschaften und Mathematik
Interaktion 1
33619 Bielefeld
Telefon: +49.521.106-7407
bernhard.bachmann@hsbi.de



Vorwort

Die Entstehungsgeschichte zu diesem Buch geht auf das Jahr 2010 zurück. Zu dieser Zeit ergab sich im Rahmen der Ausbildung von Mathematik-Studierenden die Suche nach einem anschaulichen Alltagsproblem, welches sich als Lehrbeispiel für mathematische Modellierung und Optimierung dahingehend eignet, dass die Problematik einerseits ohne spezielles Vorwissen einfach zu vermitteln ist, andererseits aber auch eine durchaus anspruchsvolle Komplexität aufweist und daher nicht ohne Weiteres zu lösen ist. Die Wahl fiel auf das *Laubharkproblem*, also die Entsorgung von herbstlichem Laub in Gärten oder Parks. Diese durch intuitiven Pragmatismus immer wieder zu lösende Alltagsproblematik ergibt sich bei genauer Analyse als ein sehr vielschichtiges und überaus komplexes Entsorgungsproblem (Näheres hierzu in Kapitel 1 und 2).

Seitdem hat sich eine Reihe von Studierenden im Bachelorstudiengang *Angewandte Mathematik* und im Masterstudiengang *Optimierung & Simulation* an der Fachhochschule Bielefeld (jüngst umbenannt in Hochschule Bielefeld – University of Applied Sciences and Arts (HSBI)) mit dieser Thematik befasst, zum Teil in Abschlussarbeiten, zum Teil in Master-Projekten. Erste Veröffentlichungen im Jahr 2016 zeigen den damaligen Zwischenstand der wissenschaftlichen Beschäftigung mit dem Laubharkproblem [13, 14]. Zwischenzeitlich bildete sich ein Projektteam, das sich – wenn auch peu à peu, so doch kontinuierlich – mit der Weiterentwicklung von Modellierungs- und Lösungsansätzen befasst. Dabei entstand ein Programm, geschrieben in MATLAB, in welchem eine Vielzahl von heuristischen Lösungsverfahren für das Laubharkproblem implementiert ist. Diese Heuristiken eignen sich für die eine oder andere Problemausprägung mehr oder weniger gut. Dieses Programm namens `lhp` ist also dazu prädestiniert, in praktischen Fällen auszuprobieren, welche Verfahrensvariante besonders geeignet ist, die konkret vorliegende Problemausprägung zu lösen. Auch soll das Programm dazu dienlich sein, Anhaltspunkte zu finden, welcher Verfahrenstyp für ähnlich gelagerte Entsorgungsprobleme einzusetzen ist, ggf. in modifizierter Form. Daher wird das MATLAB-Programm `lhp` in der gegenwärtigen Version kostenlos zur Verfügung gestellt (Näheres dazu in Abschnitt 5.3).

Jeder Leser[1] wird also aufgefordert, das Programm zu nutzen, sei es um ein konkretes Laubharkproblem zu lösen, sei es um vergleichende Gegenüberstellungen bzgl. der Lösungsgüte verschiedener Heuristiken in Bezug auf bestimmte Problemausprägungen festzustellen, sei es – hinreichende MATLAB-Kenntnisse vorausgesetzt – um das Programm zu erweitern, z.B. weitere Lösungsansätze

[1] Hier sei auf die nachfolgende Erklärung zum generischen Maskulinum verwiesen.

zu entwickeln und hinzuzufügen. Zudem gibt es noch eine Vielzahl von offenen Fragestellungen zum Laubharkproblem bzw. zu den diesbezüglichen Lösungsverfahren (hierzu siehe Kapitel 6.5).

Aus Gründen der besseren Lesbarkeit wird im Folgenden auf die gleichzeitige Verwendung weiblicher und männlicher Sprachformen verzichtet und das generische Maskulinum verwendet. Sämtliche Personenbezeichnungen (z.B. Leser, Anwender oder Entscheidungsträger) gelten gleichermaßen für alle Geschlechter.

Inhaltsverzeichnis

1 Einleitung

In diesem Buch wird exemplarisch am sog. *Laubharkproblem* als einem speziellen Problem der Entsorgungslogistik gezeigt, dass der kombinierte Einsatz von deterministischen und bionischen Verfahren zu vielversprechenden Lösungen für Ver- und Entsorgungsprobleme führen kann, wenn diese in adäquater Weise als mathematische Optimierungsmodelle formuliert werden (können).

Das jährliche Laubharken in Gärten oder Parks stellt eine sehr spezielle, alltagsnahe Ausprägung von allgemeinen Entsorgungsprozessen dar und eignet sich nicht zuletzt aufgrund dieses „Bekanntheitsgrades“ als didaktisches Musterbeispiel, um in die Bereiche der mathematischen Modellierung, Optimierung und Simulation einzuführen, zumal es sich in der Gesamtheit als ein recht komplexes Problem herausstellt. Schließlich ergeben sich hierbei mehrere Fragestellungen, die simultan oder sukzessiv zu lösen sind:

- Soll das zu entsorgende Laub auf einer größeren Garten- oder Parkfläche zu vielen kleinen oder wenigen großen Laubhaufen zusammengeharkt werden?
- Wo sollen diese Laubhaufen gebildet werden?
- Von welchen kleineren Teilflächen soll zu welchen Laubhaufen geharkt werden?
- In welcher Reihenfolge sollen dabei die einzelnen Teilflächen bearbeitet werden?
- Welcher Aufwand ergibt sich einerseits beim Harkprozess, andererseits beim Transportprozess (Abtransport der Laubhaufen mit einem Laubwagen zu einer oder zu mehreren Kompoststellen)?
- Wie lassen sich dabei unproduktive Wege möglichst vermeiden?

Diese Fragestellungen ergeben sich je nach Realproblem in mehr oder weniger ähnlicher Form auch bei anderen Ent- und Versorgungsprozessen. Aufgrund der einfachen Anschaulichkeit soll im Folgenden weiterhin das Bild eines Laubharkprozesses in einem Garten als Muster beibehalten werden. Die für dieses „Musterproblem“ entwickelten Lösungsansätze sind dann im Einzelnen auf ähnliche Problemfälle aus der Entsorgungslogistik entsprechend zu übertragen bzw. zu modifizieren. Die Bestimmung von optimalen Standorten für die Laubhaufen

erinnert an ein spezifiziertes Facility-Location-Problem oder ein Hub-Location-Problem (Näheres dazu in Abschnitt 2.4).

Einen Vorschlag für die Modellierung des Laubharkproblems findet man in [13]. Hierbei wird ein Garten durch beliebig feine Rasterung in Matrixform gebracht, wodurch endlich viele Teilflächen - im Folgenden *Felder* genannt - entstehen und die Grundlage für ein kombinatorisches Optimierungsmodell gegeben ist. Durch die Festlegung der Nachbarschaft dieser Felder, welche die Möglichkeiten von direkten Harkvorgängen beschreiben, ergibt sich auf kanonische Weise ein ungerichteter Graph, dessen Kantenbewertungen den zeitlichen Aufwand sowohl für den Hark- als auch für den Transportprozess angeben. Die Minimierung des Gesamtaufwandes, in den die Einzelaufwände für die Teilprozesse gewichtet eingehen, lässt sich somit als ein kombinatorisches Optimierungsproblem formulieren. Dieser Modellbildungsprozess wird ausführlich in Kapitel 2 behandelt. Für diejenigen Leser, die sich nicht so sehr für eine anschauliche und ausführliche Herleitung der Modellierung des Laubharkprozesses als ein kombinatorisches Optimierungsproblem interessieren, mag ein direkter Einstieg zum Ende von Kapitel 2 genügen, wo das mathematische Optimierungsmodell in konzentrierter Form vorgestellt (Abschnitt 2.3) und eine Einbettung in den Bereich der kombinatorischen Optimierung vorgenommen wird (Abschnitt 2.4).

Einige Lösungsansätze für dieses spezielle Optimierungsproblem findet man in [14]. Diese basieren zum Teil auf klassischen Heuristiken für kombinatorische Optimierungsprobleme und sind auf die spezielle Problematik passend zugeschnitten. In Kapitel 3 werden diese und einige weitere Harkstrategien vorgestellt. Hierbei handelt es sich vornehmlich um die Nachahmung von intuitiven Vorgehensweisen von fiktiven Gärtnern beim Laubentsorgungsprozess in einem Garten. All diesen Lösungsansätzen ist gemeinsam, dass es sich um *deterministische Verfahren* handelt, d.h. dass sie zu ein und demselben Problem stets dieselbe Lösung generieren, zumal sie keine zufallsbedingten Komponenten beinhalten. Insbesondere werden diejenigen deterministischen Heuristiken näher betrachtet, die später zum Gütevergleich mit den bionischen Lösungsansätzen herangezogen werden.

Ein weiteres Augenmerk in diesem Buch ist auf die Entwicklung bzw. Übertragung von *bionischen Verfahren* zur Lösung von Laubharkproblemen in Kapitel 4 gerichtet. Hierunter versteht man Lösungsansätze aus dem Bereich der *Bionik* (Biologie & Technik), welcher sich mit dem Übertragen von Phänomenen der Natur auf die Technik befasst und evolutionäre Prozesse aus der Natur auf technische oder wirtschaftliche Prozesse zwecks Optimierung überträgt. Aus der Vielzahl der in

den letzten Jahrzehnten entwickelten bionischen Verfahren[2] sind in dieser Untersuchung zwei dieser natur-inspirierten Methoden ausgewählt und speziell zur Lösung des Laubharkproblems angepasst worden. Nach einer kurzen Einführung in die allgemeinen Prinzipien bionischer Verfahren (Abschnitt 4.1) werden Anpassung und Umsetzung eines *genetischen Lösungsansatzes* in Abschnitt 4.2 und eines *Bienenverfahrens* in Abschnitt 4.3 zur Lösung von Laubharkproblemen beschrieben.

Ein Großteil der in den Kapiteln 3 und 4 vorgestellten Lösungsansätze für das Laubharkproblem ist in einem MATLAB-Programm names `lhp` implementiert worden (Kapitel 5). In Abschnitt 5.1 wird ausführlich der Aufbau des Programms mit seinen verschiedenen MATLAB-Modulen beschrieben. Im Einzelnen wird in Abschnitt 5.1.1 das Modul zur automatischen Erzeugung von Gärten vorgestellt. Der Abschnitt 5.1.2 befasst sich mit den Metadaten zur Beschreibungen der Gartenmodelle. Der Aufruf von deterministischen Heuristiken aus Kapitel 3 werden mit ihren spezifischen Parametern in Abschnitt 5.1.3 erläutert. Der Abschnitt 5.1.4 befasst sich mit der programmtechnischen Umsetzung der bionischen Lösungsverfahren aus Kapitel 4. Die Durchführung von Tests zum Vergleich der implementierten Algorithmen ist Beschreibungsgegenstand von Abschnitt 5.1.5. Dieser Abschnitt 5.1 richtet sich vornehmlich an MATLAB-Kenner, die am programmtechnischen Aufbau und den Möglichkeiten, das Programm auf der MATLAB-Command-Window-Ebene zu bedienen, interessiert sind. Darüber hinaus besteht allerdings auch die Möglichkeit, die Laubharkprobleme anhand einer grafischen Benutzeroberfläche zu bearbeiten, wozu der programmtechnische Abschnitt 5.1 lediglich als hilfreiches „Nachschlagewerk“ dienen mag.

Das Graphical User Interface (GUI) wird in Form eines Manuals in Abschnitt 5.2 ausführlich vorgestellt, wobei sich das GUI in verschiedene Bedienoberflächen aufteilt: Oberfläche zum Erstellen der Algorithmen (Abschnitt 5.2.1), zum Erzeugen und Bearbeiten von Gärten (Abschnitt 5.2.2) sowie zum Erzeugen, Verwalten und Darstellen von Tests bzw. Testergebnissen (Abschnitt 5.2.3). Damit ist es in komfortabler Weise möglich, eine Vielzahl von implementierten Lösungsverfahren an unterschiedlichsten Gärten zu testen und vergleichend gegenüberzustellen. Zudem lassen sich die einzelnen Heuristiken auf ihr Rechenzeitverhalten oder auf geeignete gartenspezifische Parametereinstellungen untersuchen, um nur einige interessante Untersuchungsmöglichkeiten anzudeuten.

[2] Einen Überblick über die Vielfalt der natur-inspirierten Methoden aus der Bionik gibt die Liste der Metaheuristiken unter https://en.wikipedia.org/wiki/Table_of_metaheuristics.

In Kapitel 6 werden die Ergebnisse einer Testreihe dargelegt, mit der die Eignung der Verfahren, die in einer großen Vielzahl verschiedener Variationsmöglichkeiten implementiert wurden, zur Lösung der Laubharkproblems untersucht worden ist. Hierzu werden zunächst die untersuchten Fragestellungen konkretisiert, wobei die Begriffe „Lösungsgüte“ und „Effizienz“ im Mittelpunkt der Betrachtungen stehen (Abschnitt 6.1). Um die Lösungsgüte und Effizienz der verschiedenen Verfahren beurteilen zu können, werden in Abschnitt 6.2 verschiedene Gütekriterien vorgestellt, die sich dahingehend unterscheiden, ob sie allein auf die primäre Ergebnisgröße *Gesamtaufwand* ausgerichtet sind und dabei auf Einzelbewertungen der Verfahren oder auf Beurteilung einer Verfahrensauswahl abzielen oder ob gleichzeitig auch die sekundäre Ergebnisgröße *Rechenaufwand* einbezogen wird und somit bikriterielle Vergleichgrößen darstellen. Die durchgeführte Untersuchung wurde anhand einer konkreten Testreihe mit ausgewählten Testgärten und unter bestimmten Parametereinstellungen vollzogen. Die dabei benutzten Festlegungen, Annahmen und Einschränkungen werden ausführlich in Abschnitt 6.3 dargelegt.

Den Kern dieses Kapitels bildet der Abschnitt 6.4, in welchem die Untersuchungsergebnisse vorgestellt und ausgewertet werden. Es werden zunächst die deterministischen Verfahren anhand der in Abschnitt 6.2 vorgestellten Gütekriterien vergleichend gegenübergestellt. Dabei stellt sich heraus, dass eine relativ kleine Auswahl der insgesamt zur Verfügung stehenden deterministischen Verfahrensvarianten ausreichend ist, um in effizienter Weise gute Lösungen zu generieren. Hierbei stützt sich die Auswahl auf sog. *komplette Überdeckungsmengen*, die je nach Parametereinstellung aus weniger als 2% aller verfügbaren Verfahrensvarianten bestehen.[3] Den 245 Grundverfahren können zwei Zusatzmodule (*Aussparung laubleerer Felder*[4], *Laubmengenobergrenzenvariation*[5]) angefügt werden, die als (mehr oder weniger effiziente) Verbesserungsstufen dienen können.[6] In Abschnitt 6.4.1.7 wird ein kritisches Zwischenfazit gezogen und eine Empfehlung ausgesprochen, welche Verfahren sich als Startlösungen für die bionischen Verfahren anbieten. In Abschnitt 6.4.2 werden die Testergebnisse zu den beiden bionischen Verfahren (*genetischer Algorithmus*, *Bienenalgorithmus*) ausgewertet und den Ergebnissen der deterministischen Verfahren vergleichend gegenübergestellt. Es stellt sich heraus, dass die bionischen Verfahren keine effiziente Alternativen zu den besten deterministischen Verfahren darstellen, wenn ihre Optimumsuche auf Basis willkürlicher Anfangslösungen erfolgt. Erwartungsgemäß

[3] Näheres hierzu siehe in Abschnitt 6.4.1.3.
[4] Näheres hierzu siehe in Abschnitt 6.4.1.5.
[5] Näheres hierzu siehe in Abschnitt 6.4.1.6.
[6] Einen Überblick über die Gesamtheit der deterministischen Verfahren liefern Tabelle 11 (Seite 100) sowie die Liste aller determinstsichen Verfahrensvarianten im Anhang A.3.

aber liefern die bionischen Verfahren verbesserte Lösungen, wenn ihnen gute Startlösungen mitgegeben werden. Ein abschließendes Resümee dazu findet man in Abschnitt 6.5.

Abschließend werden in Kapitel 7 einige offene Fragestellungen aus dem Bereich der Ver- und Entsorgungsproblematik im Allgemeinen und aus dem Problemkomplex *Laubharkoptimierung* im Speziellen angesprochen, die sich zum Teil aus den Ergebnissen der hier vorgenommenen Untersuchungen neu ergeben haben oder auch aufgrund des beschränkten Untersuchungsrahmens bislang noch ausgeklammert worden sind. Mögen diese Hinweise das Interesse von Lesern dieses Buches für die eine oder andere Fragestellung wecken und zu weiteren Untersuchungen anregen.

2 Das Laubharkproblem als kombinatorisches Optimierungsproblem

Um ein reales Laubharkproblem mit Hilfe von mathematischen Optimierungsverfahren einer Lösung zuführen zu können, bedarf es zunächst einer Abstraktion des Problems zwecks Bildung eines mathematischen Modells. Die Modellbildung erfolgt in mehreren Stufen. Zunächst wird ein abstraktes Modell für den Garten geschaffen. Anschließend werden Überlegungen zur Festlegung einer geeigneten Zielgröße angestellt und ein Aufwandsmodell entwickelt. Dabei wird die Relevanz von problemspezifischen Nebenbedingungen und Annahmen diskutiert.

2.1 Gartenmodelle

In der folgenden Abbildung ist ein kleiner Garten skizziert. Man erkennt die Rasenfläche (lindgrün), den Teich (blau), das Gerätehaus (rot), die Kompoststelle (dunkeloliv), einige Bäume (braun) und die gepflasterten Gehwege (grau).[7]

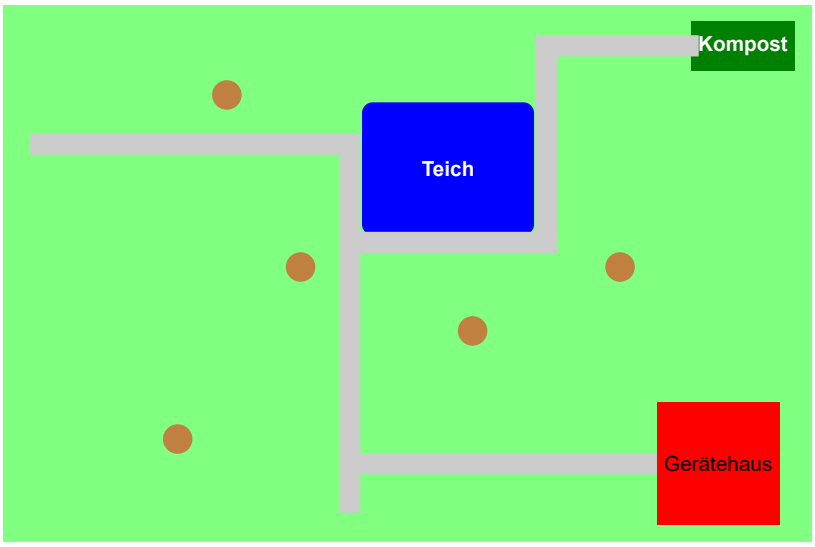


Abbildung 1: *Skizze eines Gartens*

[7] Natürlich sind Rasenflächen nicht in allen Gärten rechteckig, aber diese Gegebenheit wird die Modellierung des Gartens und die Veranschaulichung der noch zu entwickelnden Laubharkstrategien erleichtern; daher werde die Rechteckform o.B.d.A. angenommen.

Als weitergehende Abstraktionsstufe[8] bietet sich die Rasterung der Rechteckfläche an (siehe Abbildung 2). Hierbei entsteht eine Matrixform, bestehend aus m Zeilen und n Spalten. Die ***Felder*** in dieser Matrixdarstellung werden mit (i, j) bezeichnet, wobei i und j auch als ***Koordinaten*** des Feldes bezeichnet werden $(i = 1, \ldots, m,\ j = 1, \ldots, n)$.[9]

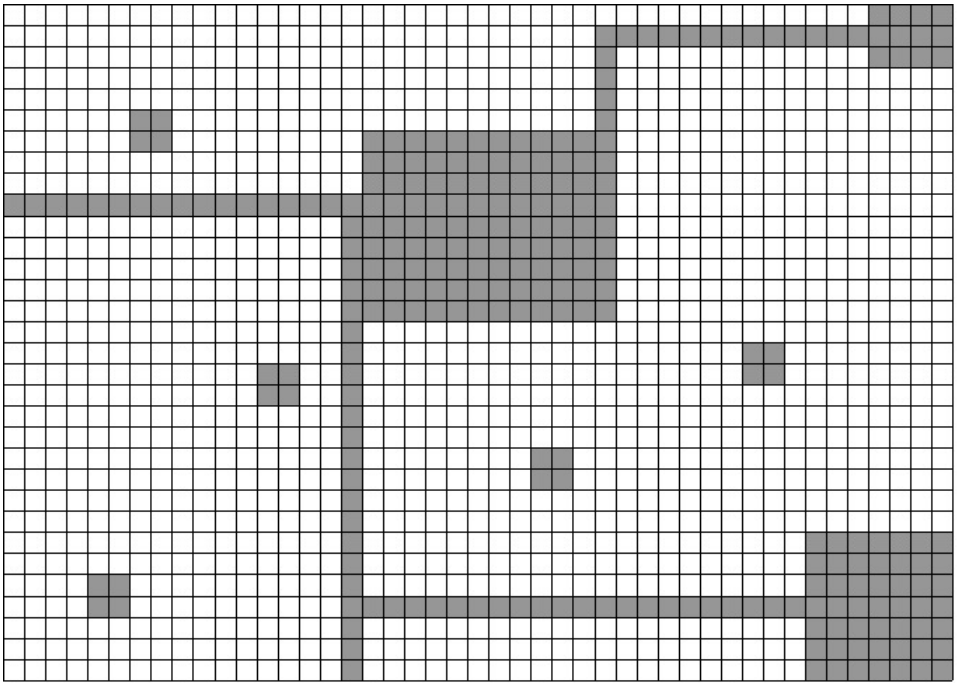


Abbildung 2: *Gerasterte Gartendarstellung*

Es wird sich als sehr praktisch erweisen, jedes Feld anstelle eines Koordinatenpaares (i, j) auch durch eine natürliche Zahl $a \in \mathbb{N}$ zu beschreiben. Hierfür wird jedem Feld (i, j) bijektiv die Zahl a zugewiesen gemäß

$$a = (i - 1) \cdot n + j. \tag{2.1}$$

Im Garten wird das Laub nun in der Regel sehr unterschiedlich auf der Rasenfläche (und den Gehwegen) verteilt liegen.[10] Genau genommen müsste man sich jetzt die Mühe machen, in jedem relevanten Feld (i, j), das zur Rasenfläche (oder zu den Gehwegen) gehört, die Anzahl der Laubblätter zu zählen und diese

[8] Als erste Abstraktionsstufe muss bereits die Skizzierung des Gartens angesehen werden, zumal diese schon ein Modell der Realität darstellt. Streng genommen wäre sogar schon ein Foto des Gartens (z.B. aus der Vogelperspektive) ein Modell.

[9] Wie fein- bzw. grobmaschig die Rasterdarstellung sein soll, bleibt dem Modellbauer überlassen. Je grober die Rasterung angelegt wird, um so mehr werden die realen Konturen des Gartens verloren gehen. Es wird also die „Kunst des Modellierens“ sein, die Größe $m \times n$ der Matrix so einzurichten, dass die Laubharkstrategien, die anhand des Rastermodells getestet werden, den realen Ablauf eines Harkprozesses in seinen wesentlichen Grundzügen zweckdienlich widerspiegeln.

[10] Hierbei muss von vornherein festgelegt sein, welche Felder vom Laub befreit werden sollen. Dieses sind sicherlich die Felder der Rasenfläche, können womöglich aber auch die Felder der Gehwege sein, während die Felder, die zum Teich, zum Gerätehaus und zum Baumbestand zählen, keiner Laubentfernung durch Harken unterstellt werden sollen bzw. können.

Anzahl dem Feld (i, j) zuzuweisen. Eine derart genaue Laubbestandsaufnahme käme allerdings einer Einzelblattlese gleich, wonach sich das Laubharkproblem nicht mehr stellte. Daher werden die Laubmengen in den Feldern höchstens nach Augenschein geschätzt und registriert. Jedes Feld a bekommt somit eine (wie auch immer ermittelte) ***Laubmenge*** $M(a)$ zugeordnet.[11] Diese Laubmenge wird in ***Laubmengeneinheiten*** [ME] gemessen.[12] Für jede Laubmenge gilt sinnvollerweise die Nichtnegativitätsbedingung:

$$M(a) \geq 0. \tag{2.2}$$

Dabei kann die Laubmenge je nach Gutdünken des Modellbauers bzw. nach Präzision der Laubmengenermittlung als reelle oder als ganze Zahl zugelassen werden. Der zweite Fall, d.h. $M(a) \in \mathbb{N}_0$, bietet sich an, wenn zur Ermittlung der Laubmengen feste Klassengrößen vorgegeben sind, etwa $M(a) = 0$ für den Fall, dass kein Laub im Feld a vorhanden ist, und $M(a) = 1, 2, 3$ usw. für den Sachverhalt, dass die Laubmenge $M(a)$ mit einem Zug bzw. mit zwei bzw. drei Zügen aus dem Feld a in ein benachbartes Feld geharkt werden kann. Der Einfachheit halber soll im Folgenden vornehmlich die ganzzahlige Variante benutzt werden.

In der Matrixdarstellung des Gartens wird nun in jedes Feld $a = (i, j)$ der Wert der zugehörigen Laubmenge $M(a) = M(i, j)$ geschrieben (siehe Abbildung 3), wobei die Gleichung $M(a) = 0$ zwei Rückschlüsse zulässt: Zum einen kann es ein problemrelevantes Feld sein, welches lediglich kein Laub aufweist; zum anderen kann es sich um ein grundsätzlich auszuschließendes Feld handeln, etwa ein Teich- oder Gerätehausfeld. Die folgende Laubmengenmatrix M stellt einen mit Laub befallenen Garten dar, allerdings handelt es sich dabei um ein vereinfachtes Beispiel ($m = 10, n = 20$). Dabei seien die grau unterlegten Nullfelder diejenigen, die im Harkprozess auszulassen sind.

[11] Die Laubmenge $M(a)$ für ein Feld $a = (i, j)$ wird bei Koordinatendarstellung mit $M(i, j)$ bezeichnet.

[12] Hierbei könnte es sich um die Anzahl der Laubblätter oder um 100 Gramm Laub oder um diejenige Laubmenge handeln, die bei einem Harkzug mit einer normierten Harke (durchschnittlich) bewegt werden kann. Hierauf wird später bei der Aufwandsberechnung für den Harkprozess noch näher eingegangen.

1	2	0	3	3	2	4	3	3	2	1	2	3	4	4	3	2	1	0	0
2	2	3	2	3	2	3	3	4	3	2	3	3	5	4	4	3	1	0	0
2	4	5	3	2	2	2	2	3	1	1	1	2	4	2	5	4	3	1	0
3	6	0	4	3	2	2	2	1	0	0	0	0	2	3	6	0	4	2	3
2	5	4	4	3	5	4	3	1	0	0	0	0	1	3	6	5	3	2	2
1	2	2	2	1	5	8	6	2	0	0	0	0	2	4	3	3	3	2	2
1	4	4	3	2	5	0	7	4	2	3	4	4	5	0	4	0	0	0	0
1	4	0	5	3	6	5	6	4	3	2	1	2	1	3	3	0	0	0	0
1	2	3	5	2	3	3	4	2	1	1	2	2	2	2	1	0	0	0	0
1	2	1	2	1	1	1	1	0	0	0	1	2	3	2	2	0	0	0	0

Abbildung 3: *Beispiel für einen Garten in Matrixdarstellung*

Um die Nullelemente in der Matrix M bezüglich ihrer jeweiligen Bedeutung (laubleeres oder auszulassendes Feld) für spätere Berechnungen anhand des Zahlenwertes unterscheiden zu können (und nicht unter Zuhilfenahme von grafischen Markierungen wie etwa Schattierungen), könnte vereinbart werden, die vom Laubharkprozess auszuschließenden Felder mit einem negativen Wert (z.B. –1) oder als Blank zu identifizieren. Eine Detaillierungsmöglichkeit von sogenannten ***blockierten Feldern*** findet man in Abschnitt 5.1.1.

In Anlehnung an das Harken in einem realen Garten wird der gesamte Harkprozess in Teilprozesse unterteilt. Und zwar wird davon ausgegangen, dass die Laubmenge $M(a)$ eines Feldes a in ein benachbartes Feld b geharkt wird $(a \neq b)$. Bei diesem Teilprozess werden die Laubmengen $M(a)$ und $M(b)$ verändert, und zwar gilt:

$$M_{neu}(b) = M_{alt}(b) + M_{alt}(a), M_{neu}(a) = 0. \tag{2.3}$$

Hierbei ist allerdings noch zu vereinbaren, was unter der „Nachbarschaft“ zwischen Feldern konkret verstanden werden soll. Hierzu bieten sich zwei Nachbarschaftsmodelle an.[13]

Zwei verschiedene Felder $a = (i, j)$ und $b = (u, v)$ heißen ***benachbart***, wenn gilt:

$$|u - i| + |v - j| = 1. \tag{2.4}$$

Bei dieser Variante (a) lässt sich eine Laubmenge $M(a)$ immer nur senkrecht oder waagerecht, also in das nächste obere oder untere bzw. in das nächste linke oder rechte Feld harken (vgl. Abbildung 4 links). Die nächste Variante (b) lässt zudem noch diagonales Harken und somit vier weitere Möglichkeiten zu (vgl. Abbildung 4 rechts):

[13] Ein weiteres interessantes Nachbarschaftsmodell wäre die Darstellung des Gartens als Wabenmuster, sodass sich auf kanonische Weise jeweils bis zu sechs Feldnachbarn ergeben. Diese Darstellungsmöglichkeit wird im Folgenden aber nicht weiter ausgeführt, zumal diese sich in den Graphenmodellen wiederfinden lässt.

$$|u - i| \leq 1 \quad \wedge \quad |v - j| \leq 1. \tag{2.5}$$

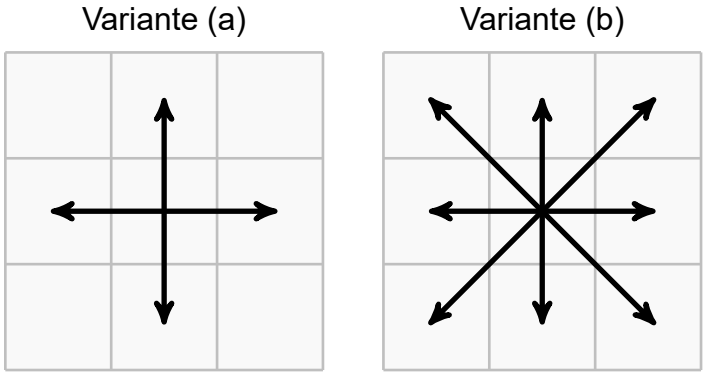


Abbildung 4: *Nachbarschaftsvarianten*

Ein in Matrixform abstrahierter Garten lässt sich anhand des Nachbarschaftsbegriffs auf kanonische Weise als ***Graph*** veranschaulichen. Jedes problemrelevante Feld a wird dabei als ein Knoten des Graphen aufgefasst; zwei benachbarte Felder a und b werden im Graph durch eine Kante $[a, b]$ verbunden. Der so entstehende Graph wird mit $G = [V, E]$ bezeichnet, wobei V die ***Knotenmenge*** und E die ***Kantenmenge*** des Graphen heißen. Da in der Praxis das Harken zwischen je zwei benachbarten Feldern a und b grundsätzlich in beide Richtungen möglich ist, handelt es sich bei G um einen ungerichteten Graph. Zudem bekommen die vom Harkprozess auszuschließenden Felder keine Knotenzuweisung. Jeder Knoten $a \in V$ von G wird mit der Laubmenge $M(a)$ des zugehörigen Feldes $a = (i, j)$ bewertet. Somit entsteht ein knotenbewerteter Graph $G = [V, E, M]$, wobei $M : V \to \mathbb{R}_{\geq}$ die ***Knotenbewertung*** von G heißt. Das in Abbildung 3 vorgestellte Gartenbeispiel in Matrixform erhält je nach Nachbarschaftsvariante die folgenden Darstellungen als Graph (siehe Abbildung 5 bzw. 6), an denen unschwer zu erkennen ist, dass somit auch beliebige Gartenformen und auch weitere Nachbarschaftsvarianten mittels Graphen abstrahiert werden können.

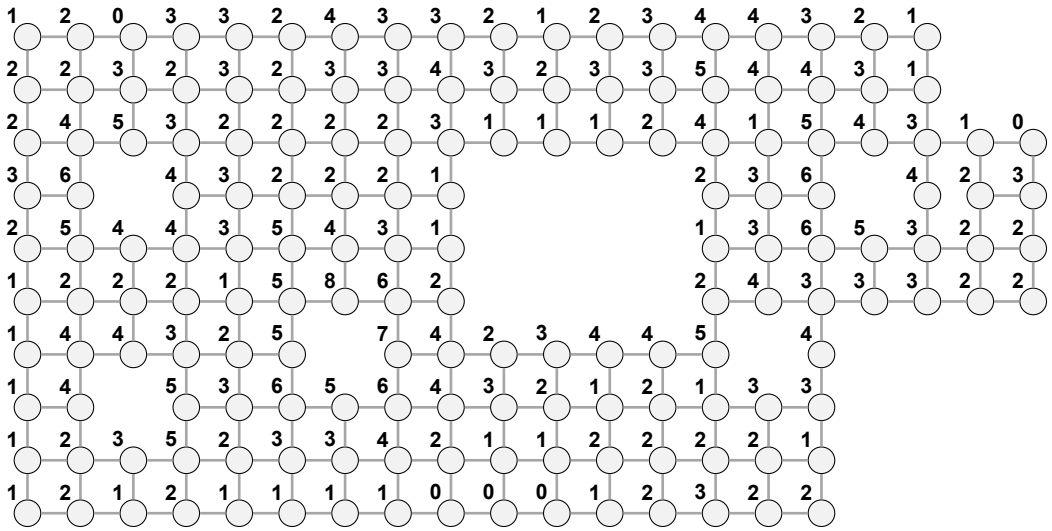


Abbildung 5: *Graphendarstellung eines Gartens gemäß Nachbarschaftsvariante (a)*

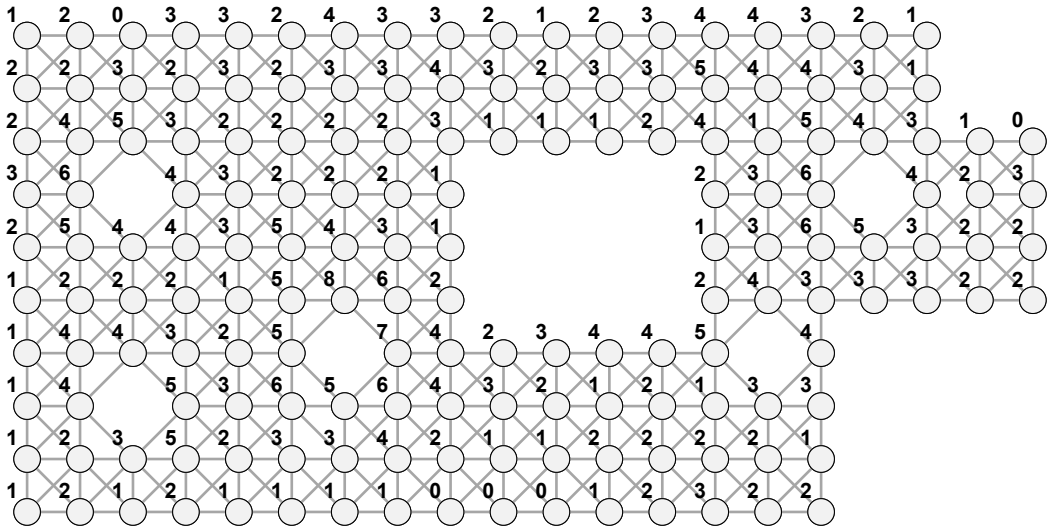


Abbildung 6: *Graphendarstellung eines Gartens gemäß Nachbarschaftsvariante (b)*

Zudem bietet es sich an, auch die Kanten des Graphen zu bewerten, wobei die Elemente $c_{a,b}$ der ***Bewertungsmatrix*** C Feldabstände oder Harkaufwände zwischen benachbarten Knoten (Feldern) a und b wiedergeben können.[14] Die Darstellung eines Gartens als knoten- und kantenbewerteter Graph hat den beträchtlichen Vorteil, dass auszulassende Felder nicht mehr in der Knotenmenge enthalten sind. Zudem lässt sich aus der Bewertungsmatrix C des Graphen sehr einfach eine Entfernungsmatrix D ermitteln, mit deren Hilfe die Hark- und Transportaufwände berechnet werden können, sobald die Hark- und Transportwege bestimmt sind.

2.2 Aufwandsberechung

Nachdem nun Matrix- und Graphenmodelle zur abstrahierten Darstellung von Gärten eingeführt sind, stellt sich die ebenso wichtige Frage nach der Zielgröße, an der die Güte eines Entsorgungsprozesses gemessen werden soll. Hier ist wieder der Entscheidungsträger gefragt, denn dieser legt fest, ob er ein möglichst Zeit, Wege oder Kräfte sparendes Vorgehen anstrebt. Um hierfür viele Möglichkeiten offen zu lassen, soll im Folgenden sehr allgemein von „***Zeitkosten***“ gesprochen werden. Diese werden in der Dimension [ZE] für ***Zeitkosteneinheit*** gemessen. Hierbei kann es sich im konkreten Fall um Zeit [t] oder Kosten [GE] oder andere Messgrößen handeln. Es wird allerdings wichtig sein, darauf zu achten, dass alle Teilprozesse in derselben Dimension gemessen werden, sodass ein eindeutiges Gütemaß für den gesamten Entsorgungsprozess zur Verfügung steht.

Der gesamte Entsorgungsprozess besteht im Wesentlichen aus den drei Teilprozessen Harken, Aufladen und Abtransportieren des Laubes. Und so, wie es ver-

[14] Näheres hierzu in Abschnitt 2.2.

schiedene Nachbarschaftsmodelle gibt, lassen sich auch verschiedene Ansätze zur Berechnung des jeweiligen Aufwandes dieser Teilprozesse finden.

2.2.1 Berechnung des Harkaufwandes

Der Harkprozess besteht aus dem Laubharken selbst, also aus Tätigkeiten, bei denen das Laub bewegt wird, aber auch aus unproduktiven Wegezeiten, die zwischen den einzelnen Harkvorgängen anfallen können. Diese werden im Folgenden getrennt voneinander betrachtet. Zunächst wird in diesem Abschnitt der eigentliche Harkprozess betrachtet; die unproduktiven Wege werden in Abschnitt 2.2.3 untersucht.

Eine sehr einfache Berechnungsvorschrift ist es, den ***Harkaufwand*** $HA(a, b)$ eines Teilprozesses, d.h. den Aufwand für das Harken der aktuellen Laubmenge $M^*(a)$ vom Feld a in das Feld b $(a \neq b)$, proportional zur Laubmenge anzunehmen:

$$HA(a, b) = M^*(a) \cdot \alpha_H, \alpha_H > 0, \tag{2.6}$$

wobei α_H als ***Harkaufwandsfaktor*** oder kurz als ***Harkparameter*** bezeichnet werden soll, welcher angibt, wie viel Zeitkosten beim Harken einer Laubmengeneinheit [ME] von einem Feld a zu einem benachbarten Feld b entsteht. Dabei wird zunächst angenommen, dass der Harkparameter unabhängig von den Feldern eine konstante Größe darstellt. Der Parameter α_H hat in diesem Fall die Dimension [ZE/ME], sodass sich für den Harkaufwand die Dimension [ZE] ergibt. Mit $M^*(a)$ ist diejenige aktuelle Laubmenge im Feld a gemeint, die zum Zeitpunkt des Harkens von Feld a in das Nachbarfeld b vorliegt; die aktuelle Laubmenge $M^*(a)$ wird sich durch den ausgeführten Harkprozess im Allgemeinen von der initialen Laubmenge $M(a)$ unterscheiden.

Nach genauerer Betrachtung dieser simplen Harkaufwandsberechnung stellen sich allerdings erste Zweifel an deren Praxisnähe ein. Letztere lässt sich nämlich wohl nur unter idealisierten Voraussetzungen vertreten. Da wäre zunächst die Annahme der Feldunabhängigkeit zu untersuchen. Ist diese in der Praxis nicht gegeben, könnte allerdings der Übergang von der konstanten Größe α_H auf variable Größen $\alpha_H(a, b)$ als erste Verallgemeinerung der Berechnung des Harkaufwandes sinnvoll sein[15]:

$$HA(a, b) = M^*(a) \cdot \alpha_H(a, b). \tag{2.7}$$

[15] Die Harkaufwandsfaktoren $\alpha_H(a, b)$ können beim Graphenmodell ohne Weiteres als Elemente der Bewertungsmatrix C hinterlegt werden.

Somit ergibt sich (2.6) auf kanonische Weise als Spezialfall von (2.7), wenn $\alpha_H(a,b) \equiv \alpha_H$ für alle benachbarten Felderpaare (a,b) gilt $(a \neq b)$. Zudem wird $\alpha_H(a,a) = 0$ für alle Felder a vereinbart.

Weiterhin ist die Annahme der Proportionalität zweifelhaft. Hierbei wird unterstellt, dass sich der Aufwand für das Harken eines Blattes bei zwei Blättern verdoppelt, wenn exemplarisch für die Laubmengeneinheit [ME] ein Laubblatt angenommen wird. Praxisnäher aber dürfte die Unterstellung sein, dass ein Zug mit einer Harke nur unwesentlich von der damit bewegten Laubmenge abhängt, insbesondere im Hinblick auf die dafür verbrauchte Zeit. Erst wenn die Laubmenge eine kritische Höhe überschreitet, sodass mehr als ein einziger Zug zum Harken nötig ist, um die Laubmenge $M^*(a)$ von Feld a nach Feld b zu bewegen, ergibt sich ein signifikanter zeitlicher Mehraufwand. Um diesem Umstand im Modell gerecht zu werden, bietet es sich an, für diese kritische Größe eine positive reelle Zahl M_k einzuführen. Ist die aktuelle Laubmenge $M^*(a)$ kleiner oder gleich diesem kritischen Wert, so wird genau ein Harkzug benötigt. Andernfalls wird man mit einem Zug nicht auskommen, sondern wird zwei oder mehrere Harkzüge benötigen. Diese reelle Zahl M_k soll ***kritische Laubharkmenge*** genannt werden.

Die Anzahl der benötigten Harkzüge bei vorliegender Laubmenge $M^*(a)$ ergibt sich dann als nächste größere ganze Zahl zum Quotienten $M^*(a)/M_k$, sodass sich der Harkaufwand gegenüber (2.7) folgendermaßen ändert[16]:

$$HA(a,b) = \left\lceil \frac{M^*(a)}{M_k} \right\rceil \cdot \alpha_H(a,b). \tag{2.8}$$

Man beachte nun aber, dass sich die Dimension der Harkparameter α_H bzw. $\alpha_H(a,b)$ geändert hat, zumal der Quotient $M^*(a)/M_k$ eine dimensionslose Größe darstellt ([ME]/[ME]). Danach müsste der Parameter α_H bzw. $\alpha_H(a,b)$ dieselbe Dimension wie der Harkaufwand $HA(a,b)$ besitzen, also in Zeitkosteneinheiten [ZE] gemessen sein. Diese Vereinbarung soll im Folgenden beibehalten werden, d.h. der Harkparameter α_H bzw. $\alpha_H(a,b)$ gibt an, wie viele Zeitkosteneinheiten für einen Harkzug benötigt werden, und zwar unabhängig von der Laubmenge, solange diese im Intervall zwischen 0 und der kritischen Laubharkmenge liegt, d.h. $0 < M^*(a) \leq M_k$.

Nachdem nun mehrere Varianten zur Modellierung des Harkaufwandes im Nachbarschaftsfall, d.h. für das Harken von einem Feld a auf ein Nachbarfeld b, vorgestellt worden sind, sollen weitergehende Überlegungen angestellt werden, wie sich diese Einzelaufwände für das Harken über mehrere Felder ($a \to b \to c \to$

[16] Hierbei sind $\lceil\,\rceil$ die Gaußklammern für die Aufrundung zur nächstgrößeren ganzen Zahl.

$d \to \ldots$) übertragen. Hierzu wird exemplarisch ein kleiner Ausschnitt (die nordwestliche Ecke) aus dem obigen Beispiel-Garten entnommen.

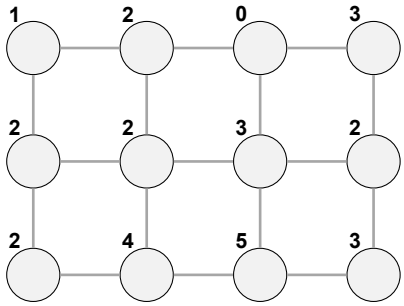


Abbildung 7: *Nordwestlicher Ausschnitt aus Abbildung 5*

Man nehme nun zunächst den idealisierten Fall an, dass die Laubmengen $M(a)$, die als Knotenbewertungen ersichtlich sind, mit der Anzahl der benötigten Harkzüge übereinstimmen. Es wird dabei also angenommen, dass jede Laubmenge $M(a)$ ein ganzzahliges Vielfaches der kritischen Laubharkmenge M_k bei konstantem Harkaufwandsparameter $\alpha_H = 1$ ist. Beispielsweise bedeutet $M(1,2) = 2$, dass die Laubmenge aus dem Feld $a = (1,2)$ mit genau zwei Harkzügen auf ein benachbartes Feld b zu harken ist. Somit ergibt sich unter Vernachlässigung der Einheiten $HA(a,b) = M(a) = 2$.

Es sei nun zunächst die Aufgabe gestellt, das gesamte Laub in diesem verkleinerten Garten auf *ein* ausgewähltes Feld zusammenzuharken, wobei sich die Frage nach dem gesamten Harkaufwand dafür stellt.

In diesem Zusammenhang stellt sich zunächst die Frage nach einer maximalen Laubmenge, die sich auf einem Feld „antürmen“ lässt und die es alleine schon aus natürlichen Stabilitätsgründen für einen Laubhaufen gibt. Eine solche ***maximale Laubmenge*** könnte für jedes Feld anders ausfallen, d.h. $M^*(a) \leq \overline{M}(a)$, oder als gemeinsame Obergrenze für alle Felder festgelegt werden, d.h. $M^*(a) \leq \overline{M}$ für alle Felder a. Im Folgenden soll nur noch der einfache Fall einer gemeinsamen maximalen Laubmenge $\overline{M}$ zur Anwendung kommen.

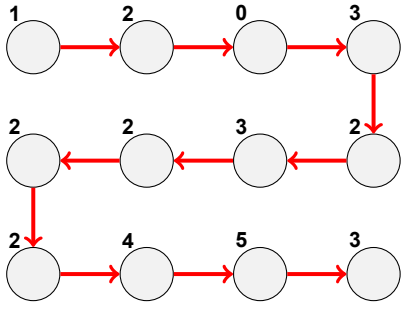


Abbildung 8: *Zickzack-Harken (Variante 1)*

Um die obige Aufgabe erfüllen zu können (also *ein* Sammelhaufen), muss die maximale Laubmenge hinreichend groß sein; im obigen Beispiel muss offensichtlich gelten: $\overline{M} \geq 29$. Andernfalls käme man mit einem einzigen Sammelhaufen nicht aus.

Exemplarisch soll zunächst eine Harkstrategie angewendet werden, wie sie in Abbildung 8 als intuitives „Zickzack“-Harken skizziert ist.

Der gesamte Harkaufwand, der sich aus den in Tabelle 1 aufgelisteten Einzelaufwänden (Anzahl der Harkzüge) zusammensetzt, beträgt dabei 124 Harkzüge.

Tabelle 1: *Harkaufwand für Zickzack-Variante 1*

Harken (von …nach)	**Aufwand** (Anzahl Harkzüge)
(1,1) → (1,2)	1
(1,2) → (1,3)	3
(1,3) → (1,4)	3
(1,4) → (2,4)	6
(2,4) → (2,3)	8
(2,3) → (2,2)	11
(2,2) → (2,1)	13
(2,1) → (3,1)	15
(3,1) → (3,2)	17
(3,2) → (3,3)	21
(3,3) → (3,4)	26
Summe	**124**

Eine alternative Zickzack-Strategie ist in Abbildung 9 skizziert. Der gesamte Harkaufwand für diese Alternative beträgt 150 Harkzüge (vgl. Tabelle 2).

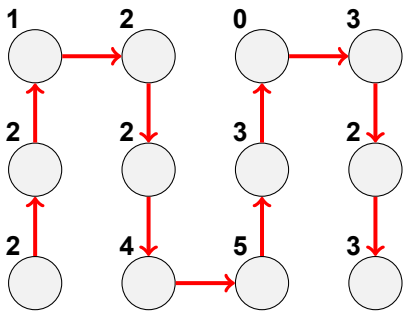


Abbildung 9: *Zickzack-Harken (Variante 2)*

Den beiden Zickzack-Strategien wird in Abbildung 10 eine weitere Harkmöglichkeit gegenübergestellt. Der gesamte Harkaufwand für diese 3. Variante beträgt 47 Harkzüge (vgl. Tabelle 3).

Tabelle 2: *Harkaufwand für Zickzack-Variante 2*

Harken (von …nach)	**Aufwand** (Anzahl Harkzüge)
(3,1) → (2,1)	2
(2,1) → (1,1)	4
(1,1) → (1,2)	5
(1,2) → (2,2)	7
(2,2) → (3,2)	9
(3,2) → (3,3)	13
(3,3) → (2,3)	18
(2,3) → (1,3)	21
(1,3) → (1,4)	21
(1,4) → (2,4)	24
(2,4) → (3,4)	26
Summe	**150**

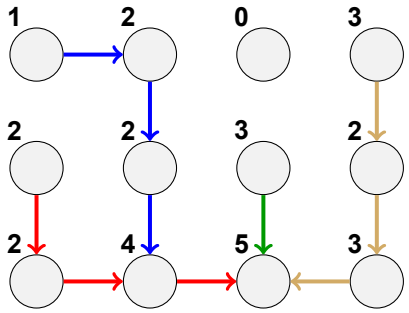


Abbildung 10: *Hark-Variante 3*

Tabelle 3: *Harkaufwand für Variante 3*

Harken (von …nach)	**Aufwand** (Anzahl Harkzüge)
(1,1) → (1,2)	1
(1,2) → (2,2)	3
(2,2) → (3,2)	5
(2,1) → (3,1)	2
(3,1) → (3,2)	4
(3,2) → (3,3)	13
(2,3) → (3,3)	3
(1,4) → (2,4)	3
(2,4) → (3,4)	5
(3,4) → (3,3)	8
Summe	**47**

Anhand dieser Beispiele zeigt sich bereits sehr deutlich, dass der gesamte Harkaufwand entscheidend von der verwendeten Harkstrategie abhängt. Dabei beeinflusst nicht nur die Festlegung der Harkreihenfolge ($a \to b \to c \to d \to \ldots$) den gesamten Harkaufwand, sondern auch die Bestimmung desjenigen Feldes, auf

dem der durch das Zusammenharken entstehende *Laubhaufen* errichtet werden soll.[17] Zudem sei bereits an dieser Stelle angemerkt, dass bei der Harkvariante 3 unproduktive Wege während des Harkprozesses entstehen, zumal dieser mehrfach unterbrochen werden muss, und zwar jeweils beim „Farbwechsel“ der Teilharkprozesse; diese unproduktiven Wege nehmen Zeit in Anspruch und sind somit dem produktiven Harkaufwand zuzuschlagen. Demgegenüber treten hier bei den Zickzack-Strategien keine solchen unproduktiven Wege auf. Näheres hierzu wird noch in Abschnitt 2.2.3 diskutiert werden.

2.2.2 Darstellung der Harkprozesse durch Nachfolgerfunktionen

Um später zur Entwicklung von effizienten Strategien auf Verfahren aus der kombinatorischen Optimierung zurückgreifen zu können, wird vorweg ein Darstellungsmodell für Harkprozesse eingeführt, an welchem die Berechnung des jeweiligen Harkaufwands in einheitlicher Form vollzogen werden kann.

Als eine Lösung des Laubharkproblems gilt eine explizite ***Harkvorschrift*** in Form einer Abbildung $s : V \to V$; hierbei wird jedem Feld $a \in V$ dasjenige Nachbarfeld $b = s(a) \in V$ zugewiesen, wohin die aktuelle Laubmenge $M^*(a)$ geharkt wird. Eine anschauliche Darstellung einer Harkvorschrift ist die Tabellenform, wobei die Nummerierung der Felder gemäß (2.1) gilt (vgl. Abbildung 10):

Tabelle 4: *Beispiel einer Harkvorschrift mittels Nachfolgerfunktion*

a	1	2	3	4	5	6	7	8	9	10	11	12
$s(a)$	2	6	3	8	9	10	11	12	10	11	11	11

Die obige Tabelle besagt, dass vom Feld 1 zum Feld 2, weiter zum Feld 6, von dort zum Feld 10 und schließlich zum Feld 11 geharkt wird. Wegen $s(11) = 11$ wird im Feld 11 ein Laubhaufen zusammengeharkt; solche Felder werden im Folgenden ***Haufenfelder*** oder ***Hubs*** genannt. Zudem wird vom Feld 4 zum Feld 8, weiter zum Feld 12 und schließlich zum Haufenfeld 11 geharkt. Die Laubmenge von Feld 5 wird zum Feld 9, weiter zum Feld 10 und schließlich zum Feld 11 geharkt. Letztlich wird noch vom Feld 7 zum Feld 11 geharkt. Das Feld 3 ist wegen $s(3) = 3$ ebenfalls ein Haufenfeld (Hub), allerdings wird diesem Feld kein Beitrag von einem anderen Feld zugeliefert; es stellt somit ein *isoliertes Haufenfeld* dar.

[17] In größeren Gärten bestimmen auch noch weitere Kenngrößen die Suche nach effizienten Laubharkstrategien, z.B. die Anzahl und die Größe der zu errichtenden Laubhaufen.

Damit eine Abbildung $s : V \to V$ eine Harkvorschrift darstellt, wobei ausschließlich entlang von Kanten des Graphen G geharkt wird, muss für jeden Knoten $a \in V$ die ***Nachbarschaftsbedingung*** gelten:

$$a \neq s(a) \Rightarrow [a, s(a)] \in E. \tag{2.9}$$

Man mache sich klar, dass durch eine Abbildung $s : V \to V$, die der Nachbarschaftsbedingung genügt, auf kanonische Weise ein Digraph $\vec{G}_s = \langle V, \vec{E}_s \rangle$ induziert wird, wobei die Pfeilmenge $\vec{E}_s$ aus den gerichteten Kanten $\langle a, s(a) \rangle$ mit $a \neq s(a)$ besteht. Um ein sinnvolles Harken zu gewährleisten, muss dieser Digraph $\vec{G}_s$ zudem zyklenfrei sein.

Eine solche Abbildung $s : V \to V$, die sowohl der Nachbarschaftsbedingung (2.9) genügt als auch einen zyklenfreien Digraph $\vec{G}_s = \langle V, \vec{E}_s \rangle$ mit $\vec{E}_s = \{\langle a, s(a) \rangle \mid a \in V \wedge a \neq s(a)\}$ induziert, wird im Folgenden ***Nachfolgerfunktion*** genannt.[18]

Eine Harkvorschrift in Form einer Nachfolgerfunktion stellt auf kanonische Weise eine Lösung des Laubharkproblems dar, denn es werden die generierten Laubhaufen hinsichtlich der Anzahl, Lage und kumulierten Laubmenge eindeutig festgelegt. Zur Überprüfung, ob eine Abbildung $s : V \to V$ eine Nachfolgerfunktion ist, kann ein einfaches Prüfmodul implementiert werden.

Man mache sich ebenfalls klar, dass der durch eine Nachfolgerfunktion s induzierte Digraph $\vec{G}_s$ eine „inverse Baumstruktur“ besitzt, d.h. der inverse Digraph ist ein gerichteter Wald. Im obigen Beispiel ergibt sich ein gerichteter Wald mit zwei Wurzelbäumen, wobei der eine Baum den Wurzelknoten 11 und die Blätter 1, 4, 5 und 7 hat, der zweite Baum allein aus dem isolierten Knoten 3 besteht (siehe Abbildung 11).

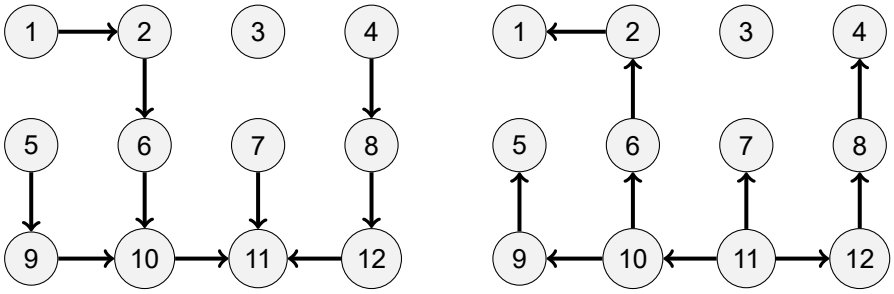


Abbildung 11: *Durch Nachfolgerfunktion induzierter Digraph (links) und Wald (rechts)*

Die Quellen[19] des durch eine Nachfolgerfunktion s induzierten Digraphen $\vec{G}_s$ werden im Folgenden auch ***Harkquellen*** genannt; sie bilden die ***Harkquellenmenge*** $Q(s)$ von s. Die Senken[20] von $\vec{G}_s$ stellen die Haufenfelder (Hubs) des Harkpro-

[18] Vgl. [13], S. 82 .
[19] Quellen sind Knoten ohne Pfeileingänge in einem Digraph.
[20] Senken sind Knoten ohne Pfeilausgänge in einem Digraph.

zesses dar; sie bilden die ***Hubmenge*** $Hub(s)$ von s.

Um zu gewährleisten, dass eine Nachfolgerfunktion s nicht nur im Initialzustand, sondern auch während des gesamten Harkprozesses vorgegebene Kapazitätsbeschränkungen $\overline{M}(a)$ für alle Felder $a \in V$ erfüllt, wird noch ein Zulässigkeitsbegriff eingeführt: Eine Nachfolgerfunktion s wird ***zulässig*** bzgl. $\overline{M}$ genannt, wenn bei dem durch s induzierten Harkprozess beachtet wird, dass in keinem Feld a und erst recht in keinem Haufenfeld mehr Laub angehäuft wird, als gemäß maximaler Laubmenge $\overline{M}(a)$ bzw. $\overline{M}$ erlaubt ist. Um diesen Sachverhalt zu formalisieren, wird zunächst zu jedem Knoten a des Digraphen $\vec{G}_s$ die ***Erreichbarkeitsmenge*** $R(a)$ bestimmt; diese enthält alle Knoten $i \in V$, von denen aus der Knoten a entlang einer Pfeilfolge in $\vec{G}_s$ erreichbar ist. Die Abbildung $M_s^* : V \to \mathbb{R}_{\geq}$ mit der Bildungsvorschrift

$$M_s^*(a) := \sum_{k \in R(a)} M(k) \text{ für alle Knoten } a \in V \tag{2.10}$$

heißt die durch s induzierte ***Anhäufungsfunktion***. Eine Nachfolgerfunktion s heißt dann ***zulässig*** bzgl. $\overline{M}$, wenn gilt:

$$M_s^*(a) \leq \overline{M}(a) \text{ für alle Knoten } a \in V. \tag{2.11}$$

Es ist leicht einzusehen, dass im konstanten Falle (d.h. $\overline{M}(a) = \overline{M}$ für alle $a \in V$) die Zulässigkeitsprüfung allein auf die Hubs $a \in Hub(s)$ eingeschränkt werden kann, zumal sich daraus die Zulässigkeit für alle Knoten ergibt. Umgekehrt (d.h. $\overline{M}(a)$ nicht konstant) kann es Fälle geben, dass die Bedingung (2.11) zwar für alle Hubs, nicht aber für alle sonstigen Knoten gelten muss.

Zur effizienten Bestimmung des Harkaufwandes[21] anhand von (zulässigen) Nachfolgerfunktionen wird eine einfache Rechenprozedur benutzt: Zu gegebener (zulässiger) Nachfolgerfunktion s eines Graphen $G = [V, E]$ wird die Anhäufungsfunktion M_s^* ermittelt. Danach wird zu jedem Nicht-Hub $a \in V \setminus Hub(s)$ der Harkaufwand $HA(a, s(a))$ gemäß einer der Harkaufwandsformeln (2.6) - (2.8) bestimmt. Abschließend werden die Einzelaufwände aufsummiert:

$$HA_{\Sigma}(s) = \sum_{a \in V \setminus Hub(s)} HA(a, s(a)). \tag{2.12}$$

Für die Berechnung des Harkaufwandes einer Harkvorschrift ist die Bestimmung derjenigen Felder relevant, von denen aus der jeweils nächste Harkzug gemacht werden soll. Solche Felder sollen als *aktuelle Harkfelder* bezeichnet werden. Ein

[21] Man beachte, dass hier zunächst der reine Harkaufwand ohne Berücksichtigung von unproduktiven Wegen gemeint ist.

aktuelles Harkfeld zeichnet sich dadurch aus, dass sämtliche Vorgänger im Verlaufe des Verfahrens bereits leer geharkt worden sind.[22]

Beispielsweise ergibt sich für die obige Nachfolgerfunktion s (vgl. Tabelle 4) ein Harkaufwand $HA_\Sigma(s)$ gemäß (2.6) mit $\alpha_H = 1$ von 47 ZE. Demgegenüber ergibt sich ein Harkaufwand von 124 ZE, falls der gemäß Nachfolgerfunktion s^* induzierte Harkprozess durchgeführt wird (vgl. Tabelle 5).

Tabelle 5: *Beispiel einer alternativen Nachfolgerfunktion*

a	1	2	3	4	5	6	7	8	9	10	11	12
$s^*(a)$	2	3	4	8	9	5	6	7	10	11	12	12

Beide Harkprozesse erzeugen jeweils *einen* Laubhaufen mit einer kumulierten Laubmenge 29 ME, allerdings an verschiedenen Stellen (Feld 11 bzw. 12 ist der jeweilige Hub). Entsprechend können sich die jeweils anschließenden Transportprozesse in ihren Transportaufwänden unterscheiden, ebenso die unproduktiven Wege. An diesem Beispiel wird allerdings deutlich, dass bei festgelegter Berechnungsgrundlage jeder vorgegebenen Nachfolgerfunktion s der kumulierte Harkaufwand $HA_\Sigma(s)$ zugewiesen werden kann.

In folgender Tabelle sind einige verschiedene Nachfolgerfunktionen aufgelistet. Es handelt sich dabei allerdings um solche „Spezialfälle“, bei denen stets *genau ein* Laubhaufen erzeugt wird.

Tabelle 6: *Ausgewählte Nachfolgerfunktionen mit zugehörigem Harkaufwand*

a	1	2	3	4	5	6	7	8	9	10	11	12	$HA_\Sigma(s_i)$
$s_1(a)$	2	6	3	8	9	10	11	12	10	11	11	11	47
$s_2(a)$	2	3	4	8	9	5	6	7	10	11	12	12	124
$s_3(a)$	2	3	4	4	1	2	3	4	5	6	7	8	76
$s_4(a)$	2	3	4	4	6	7	8	4	10	11	12	8	76
$s_5(a)$	2	6	3	8	6	7	7	7	5	11	7	8	**46**
$s_6(a)$	5	6	7	8	6	10	11	7	10	10	10	11	50
$s_7(a)$	5	6	7	8	6	7	8	8	5	6	7	8	59
$s_8(a)$	2	3	4	4	6	7	8	4	5	6	7	8	76

[22] Diese Vorgehensweise entspricht dem Prinzip des sukkzessiven Abpflückens eines gerichteten Baumes mit dem Hubknoten als Wurzel bzw. dem inversen Vorgehen bei einer topologischen Sortierung von zyklenfreien Digraphen. Allerdings bleibt anzumerken, dass bei dieser Harkaufwandberechnung die Zeitkosten für die unproduktiven Wege zwischen den einzelnen Feldern, insbesondere beim Wechsel von Teilharkprozessen, noch unberücksichtigt bleiben. Auf eine entsprechende Ergänzung der Harkaufwandberechnung durch Einbeziehung dieser Wegaufwände wird auf den folgenden Abschnitt 2.2.3 verwiesen.

Es dürfte unverkennbar sein, dass die Anzahl der verschiedenen Nachfolgerfunktionen, insbesondere bei freier Wahl der Laubhaufenanzahl, schon bei „kleinen“ Graphen sehr groß ausfällt und mit der Größe des Graphen enorm anwächst. Somit besteht eine komplexe Aufgabe im Wesentlichen darin, durch geeignete Harkstrategien zulässige Nachfolgerfunktionen zu generieren und ausgehend von einer zulässigen Nachfolgerfunktion bessere Nachfolgerfunktionen in der „Nachbarschaft“ zu finden, um damit das Laubharkproblem schrittweise zu lösen.

2.2.3 Berechnung von unproduktiven Wegezeiten

Es ist offensichtlich, dass durch die Nachfolgerfunktion s zwar einzelne Harkschritte und auch die grundsätzliche „Harkstruktur“ (Anzahl und Lage der Haufenfelder mit kumulierter Laubmenge) festgelegt werden, aber der gesamte Harkvorgang dennoch nicht eindeutig beschrieben wird, weil die Reihenfolge der einzelnen Harkschritte dadurch noch nicht festgelegt ist. Die folgende Darstellung der Nachfolgerfunktion s (vgl. Tabelle 7) lässt offen, in welchem Feld der Harkprozess beginnen soll, wie nach einem Harkschritt $a \to s(a)$ fortgesetzt werden soll, insbesondere dann, wenn $s(a)$ noch Vorgängerfelder hat, die nicht leer geharkt sind, oder $s(a)$ ein Haufenfeld darstellt.

Tabelle 7: *Beispiel einer Nachfolgerfunktion*

a	1	2	3	4	5	6	7	8	9
$s(a)$	4	5	6	5	8	5	7	7	8

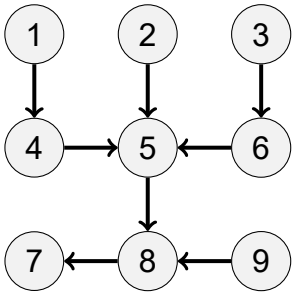


Abbildung 12: *Digraph $\vec{G}_s$ zur Nachfolgerfunktion s*

Unter Beachtung des effizienten Harkprinzips, dass nur von Feldern geharkt werden sollte, deren Vorgängerfelder bereits leer geharkt sind, wird folgerichtig mit einer Harkquelle begonnen, und bei jeder zwangsläufigen Unterbrechung des produktiven Harkprozesses wird wieder bei einer noch nicht einbezogenen Harkquelle angesetzt. Für das obige Beispiel ergeben sich anhand der vier vorhandenen Harkquellen[23] $4! = 24$ Möglichkeiten für effiziente Harkreihenfolgen.[24] In der folgenden Darstellung werden exemplarisch nur die vom Startknoten 1

[23] Die Menge der Harkquellen ist $Q(s) = \{1, 2, 3, 9\}$; vgl. Abbildung 12.
[24] Man beachte, dass alle diese Harkreihenfolgen denselben Harkaufwand $HA_\Sigma(s)$ hervorrufen.

ausgehenden Harkreihenfolgen aufgeführt; dabei werden produktive Harkschritte durch $\rightarrow$ und unproduktive Wege durch $\Rightarrow$ gekennzeichnet:

Nr. 1: $1 \rightarrow 4 \rightarrow 5 \Rightarrow 2 \rightarrow 5 \Rightarrow 3 \rightarrow 6 \rightarrow 5 \rightarrow 8 \Rightarrow 9 \rightarrow 8 \rightarrow 7$
Nr. 2: $1 \rightarrow 4 \rightarrow 5 \Rightarrow 2 \rightarrow 5 \Rightarrow 9 \rightarrow 8 \Rightarrow 3 \rightarrow 6 \rightarrow 5 \rightarrow 8 \rightarrow 7$
Nr. 3: $1 \rightarrow 4 \rightarrow 5 \Rightarrow 3 \rightarrow 6 \rightarrow 5 \Rightarrow 2 \rightarrow 5 \rightarrow 8 \Rightarrow 9 \rightarrow 8 \rightarrow 7$
Nr. 4: $1 \rightarrow 4 \rightarrow 5 \Rightarrow 3 \rightarrow 6 \rightarrow 5 \Rightarrow 9 \rightarrow 8 \Rightarrow 2 \rightarrow 5 \rightarrow 8 \rightarrow 7$
Nr. 5: $1 \rightarrow 4 \rightarrow 5 \Rightarrow 9 \rightarrow 8 \Rightarrow 2 \rightarrow 5 \Rightarrow 3 \rightarrow 6 \rightarrow 5 \rightarrow 8 \rightarrow 7$
Nr. 6: $1 \rightarrow 4 \rightarrow 5 \Rightarrow 9 \rightarrow 8 \Rightarrow 3 \rightarrow 6 \rightarrow 5 \Rightarrow 2 \rightarrow 5 \rightarrow 8 \rightarrow 7$

Wird eine Entfernungsmatrix $D = (d_{ij})_{i,j=1,\ldots,9}$ gemäß Manhattan-Metrik zu Grunde gelegt, ergeben sich für die obigen sechs Reihenfolgemöglichkeiten unterschiedliche Gesamtlängen für unproduktive Wege, wobei in diesem Beispiel der Einfachheit halber für die Kantenbewertungen $c_{ij} \equiv 1$ angenommen wird:

Nr. 1: $d_{52} + d_{53} + d_{89} = \mathbf{4}$
Nr. 2: $d_{52} + d_{59} + d_{83} = 6$
Nr. 3: $d_{53} + d_{52} + d_{89} = \mathbf{4}$
Nr. 4: $d_{53} + d_{59} + d_{82} = 6$
Nr. 5: $d_{59} + d_{82} + d_{53} = 6$
Nr. 6: $d_{59} + d_{83} + d_{52} = 6$

Somit ist die Minimierung des Aufwandes für unproduktive Wege bei einem durch eine Nachfolgerfunktion s beschriebenen Harkprozess auf die Bestimmung einer optimalen Harkquellenreihenfolge zurückgeführt. Es hat sich gezeigt [9], dass bei der Bestimmung einer „guten“ Harkquellenreihenfolge das Greedy-Verfahren „Nächster Nachfolger“ effiziente Lösungen anbietet. Hiernach wird ausgehend von einem beliebigen Startknoten $a \in V$ die nächstgelegene Harkquelle k bestimmt:[25]

$$\text{Wähle } k \in Q(s) \text{ mit } d_{ak} = \min\{d_{aj} \mid j \in Q(s)\}. \tag{2.13}$$

Nach diesem Prinzip wird jede weitere Harkquelle bestimmt, sobald der produktive Harkprozess unterbrochen werden muss, weil der jeweilige aktuelle Knoten ein Hub ist oder Vorgängerknoten besitzt, die noch nicht leer geharkt sind. Dabei wird jede Harkquelle genau einmal in Anspruch genommen und anschließend aus der Harkquellenmenge $Q(s)$ entfernt.[26] Entsprechend der ermittelten Reihenfolge der Harkquellen $q_1, q_2, \ldots, q_Q$ werden die Entfernungen zwischen dem jeweils ak-

[25] Falls der Startknoten a selbst eine Harkquelle ist, gilt $k = a$ wegen $d_{aa} = 0$.
[26] Die Mächtigkeit der durch s induzierten Harkquellenmenge $Q(s)$ werde mit $Q := |Q(s)|$ bezeichnet.

tuellen Knoten a_{i-1}, an dem der produktive Harkprozess unterbrochen wird, und der jeweils neuen Harkquelle q_i kumuliert, beginnend mit einem Startknoten a_0:

$$W(s) = \sum_{i=1}^{Q} d_{a_{i-1},q_i}. \tag{2.14}$$

Die Gesamtlänge $W(s)$ aller unproduktiven Wege $a_{i-1} \Rightarrow q_i$ für $i = 1, \ldots, Q$ werden noch mit einem ***Wegeaufwandsfaktor*** α_W gewichtet:

$$WA_\Sigma(s) = \alpha_W \cdot W(s) = \alpha_W \cdot \sum_{i=1}^{Q} d_{a_{i-1},q_i}. \tag{2.15}$$

Der Wegeaufwandsfaktor α_W hat die Dimension „Zeitkosteneinheit pro Längeneinheit“ $[ZE/LE]$, wenn die Entfernungen d_{jk} bzw. die Kantenbewertungen c_{jk} nach Längeneinheiten [LE] gemessen werden. Als Verallgemeinerung eines konstanten Faktors α_W können auch von den jeweiligen Entfernungen d_{jk} abhängige Wegeaufwandsfaktoren $\alpha_W(j,k)$ eingeführt werden; dann ändert sich die Formel (2.15) in

$$WA_\Sigma(s) = \sum_{i=1}^{Q} \alpha_W(a_{i-1}, q_i) \cdot d_{a_{i-1},q_i}. \tag{2.16}$$

Um die Berechnung des unproduktiven Wegeaufwandes einerseits und gleichzeitig auch des produktiven Harkaufwandes gemäß (2.12) andererseits aus der tabellarischen Darstellung einer Nachfolgerfunktion s unmittelbar „ablesbar“ zu machen, wird die Nachfolgerfunktion s mittels einer Permutation $\pi : V \leftrightarrow V$ ausgerichtet, indem der Wert $\pi(a)$ angibt, in welcher Spalte der neu ausgerichteten Tabelle das Paar $\left|\begin{matrix} a \\ s(a) \end{matrix}\right|$ positioniert wird. Der durch π hervorgerufene Spaltentausch beginnt mit einem Quellknoten a_1, bei dem der Harkprozess beginnen soll, d.h. $\pi(a_1) = 1$. Danach wird vom jeweils aktuellen Knoten a_ℓ gemäß s zum Knoten $s(a_\ell)$ geharkt, wenn a_ℓ weder Haufenfeld ist noch nichtleere Vorgängerknoten besitzt; dabei wird $\pi(a_\ell) = \ell$ gesetzt und $a_{\ell+1} = s(a_\ell)$ als nächster aktueller Knoten betrachtet. Umgekehrt wird im Falle einer Harkunterbrechung eine jeweils neue Harkquelle $q \in Q(s)$ als nächster aktueller Knoten gewählt, d.h . $a_{\ell+1} = q$.

In der folgenden Tabelle ist die Nachfolgerfunktion s aus Tabelle 7 an der obigen Reihenfolgemöglichkeit Nr. 3 ausgerichtet. Anhand dieser Tabelle lässt sich die Harkaufwandsberechnung sowohl der produktiven Harkschritte ($\downarrow$) als auch der unproduktiven Wege ($\nearrow$) einfach ablesen, wobei durch $\nearrow$ und $\downarrow$ bewertungsfreie Reihenfolgefortsetzungen angedeutet werden:

Tabelle 8: *Beispiel einer ausgerichteten Nachfolgerfunktion*

$\pi(a)$	1	2	3	4	5	6	7	8	9
a	1	4	3	6	2	5	9	8	7
$s(a)$	4	5	6	5	5	8	8	7	7

Tabelle 9: *Auswertung einer ausgerichteten Nachfolgerfunktion*

a	1	4	3	6	2	5	9	8	7
	↓ ↗	↓ ↗	↓ ↗	↓ ↗	↓ ↗	↓ ↗	↓ ↗	↓ ↗	↓
$s(a)$	4	5	6	5	5	8	8	7	7

Entlang dieser Sägezahn-Pfeilfolge werden bei jedem schwarzen Pfeil ↓ die produktiven Harkaufwände $HA(a, s(a))$ mit $a \neq s(a)$ und bei jedem rotem Pfeil ↗ die unproduktiven Wegeaufwände $\alpha_W \cdot d_{s(\pi^{-1}(i)),\pi^{-1}(i+1)}, i = 1, \ldots, n-1$, kumuliert. An obigem Beispiel ergibt sich die folgende Rechnung:

$$HA(1,4) + HA(4,5) + \alpha_W(5,3) \cdot d_{5,3} + HA(3,6) + HA(6,5) + \alpha_W(5,2) \cdot d_{5,2} + HA(2,5) + HA(5,8) + \alpha_W(8,9) \cdot d_{8,9} + HA(9,8) + HA(8,7).$$

Das Verhältnis zwischen den beiden Gewichtungsfaktoren α_H und α_W ist grundsätzlich beliebig. Im Bild des Laubharkens in einem Garten aber dürfte der produktive Harkaufwand ein signifikant größeres Gewicht haben, sodass eine Setzung $\alpha_W < \alpha_H$ oder gar $\alpha_W \ll \alpha_H$ realistisch ist. Es kann aber auch dann noch ein logischer Widerspruch dadurch entstehen, dass der produktive Harkaufwand zwischen zwei benachbarten Feldern a und b geringer als der entsprechende unproduktive Wegeaufwand ausfällt, etwa im Fall $M^*(a) = 0$, wo unter der Annahme $b = s(a)$ zwei Interpretationen möglich sind: Der Übergang $a \to b$ wird als „Nullmengen-Harkschritt“ aufgefasst, der keinen Harkaufwand erzeugt, oder aber als ein unproduktiver Weg, für den gleichwohl Wegeaufwand berechnet wird. Auch im Fall $\alpha_W = \alpha_H$ und $M^*(a) \leq M_k$ bzw. $\left\lceil \frac{M^*(a)}{M_k} \right\rceil \in \{0, 1\}$ wäre der unproduktive Wegeaufwand für den Schritt von a nach b gleichbedeutend mit dem Harkaufwand von a nach b. Daher wird der bisherige Harkaufwand (Abschnitt 2.2.1) um einen produktiven Wegeaufwand $\alpha_W \cdot d(a, b)$ ergänzt, sodass sich die Harkaufwandsformeln (2.7) und (2.8) folgendermaßen „verfeinern“ lassen:

$$HA(a,b) = M^*(a) \cdot \alpha_H(a,b) + \alpha_W \cdot d(a,b) \tag{2.17}$$

bzw.

$$HA(a,b) = \left\lceil \frac{M^*(a)}{M_k} \right\rceil \cdot \alpha_H(a,b) + \alpha_W \cdot d(a,b). \tag{2.18}$$

2.2.4 Berechnung des Transportaufwandes

Für das Aufwandsmodell des gesamten Laubharkproblems wird nun noch der Transportprozess einbezogen. Hierbei spielen zurückzulegende Transportwege zwischen den Hubs und dem Kompostfeld K die entscheidende Rolle.

Zu diesem Zweck wird die bisherige Entfernungsmatrix $D = (d_{ij})_{i,j \in V}$ um eine Zeile erweitert, welche die Entfernungen zwischen den Knoten $v \in V$ und dem Kompostknoten K angibt. Die erweiterte Entfernungsmatrix wird mit $D^* = (d^*_{ij})_{i,j \in V \cup \{K\}}$ bezeichnet.

Der Transportaufwand ließe sich dann proportional zu den zurückgelegten Weglängen bestimmen. Bei einer Pendeltour zwischen dem Kompostfeld K und einem Hub h ergibt sich im einfachsten Fall der Transportaufwand

$$TA(h) = 2 \cdot d^*_{h,K} \cdot \alpha_T, \tag{2.19}$$

wobei α_T als ***Transportaufwandsfaktor*** oder kurz als ***Transportparameter*** bezeichnet werden soll. Der Faktor α_T hat in diesem Fall die Dimension [ZE/LE], sodass sich für den Transportaufwand $TA(h)$ die Dimension [ZE] ergibt.

Allerdings können auch noch zusätzliche Aufwände eingerechnet werden, etwa die von der Laubmenge abhängige Auflade- und Abladezeit (*variabler Ladeaufwand*) oder ein fixer Zeitaufwand (etwa für das Vorbereiten zum Aufladen oder zum Abladen), welcher unabhängig von der Entfernung und der Laubmenge ist (*fixer Ladeaufwand*):

$$TA(h) = 2 \cdot d^*_{h,K} \cdot \alpha_T + \gamma \cdot M^*_s(h) + \sigma. \tag{2.20}$$

Hierbei geht der Parameter γ in den variablen Aufwandsanteil für das jeweilige Auf- und Abladen der Laubmenge $M^*(h)$ ein und hat die Dimension [ZE/ME], während die Konstante σ für den fixen Aufwandsanteil steht und die Dimension [ZE] hat.[27]

Bislang wird unterstellt, dass die gesamte Laubmenge $M^*(h)$, die vom Hub h abgeholt wird, auch komplett vom Transportmittel (Laubwagen oder Schubkarre) aufgenommen werden kann. Allerdings kann es vorkommen, dass die vom

[27] Eine Abhängigkeit der Parameter γ und σ von den Hubfeldern h als mögliche Verallgemeinerung, also $\gamma(h)$ und $\sigma(h)$, wird hier nicht weiter betrachtet. Es sei allerdings darauf hingewiesen, dass bei der späteren Aufsummierung der einzelnen Transportaufwände $TA(s)$ über alle Hubs (siehe (2.23)) ein konstanter Parameter γ keinen maßgeblichen Einfluss auf die Lösungsgüte einer Harkstrategie (bzw. der dadurch erzeugten Nachfolgerfunktion s) ausübt, wohl aber ein konstanter Parameter σ, da die Anzahl der durch s induzierten Hubs hierbei eine Rolle spielt.

Hub h abzuholende Laubmenge das ***maximale Ladevolumen*** $\overline{T}$ der Schubkarre überschreitet. In diesem Falle sind mehrere Pendeltouren nötig, um Hub h vom Laubhaufen zu befreien. Entsprechend ändert sich der Transportaufwand für die Laubentsorgung des Haufenfeldes h:

$$TA(h) = \left\lceil \frac{M_s^*(h)}{\overline{T}} \right\rceil \cdot 2 \cdot d_{h,K}^* \cdot \alpha_T \tag{2.21}$$

bzw.

$$TA(h) = \left\lceil \frac{M_s^*(h)}{\overline{T}} \right\rceil \cdot 2 \cdot d_{h,K}^* \cdot \alpha_T + \gamma \cdot M_s^*(h) + \left\lceil \frac{M_s^*(h)}{\overline{T}} \right\rceil \cdot \sigma. \tag{2.22}$$

Liegen mehrere Laubhaufen zum Abholen bereit, ergibt sich der gesamte Transportaufwand als die Summe aller Pendeltouren zu allen Hubs (bzgl. Nachfolgerfunktion s):

$$TA_\Sigma(s) = \sum_{h \in Hub(s)} TA(h). \tag{2.23}$$

Etwas komplizierter wird die Transportaufwandsberechnung dann, wenn zusätzlich zu den Pendeltouren auch Sammeltouren in Frage kommen. Dies hat allerdings nur dann einen Sinn, wenn die auf einer Sammeltour angefahrenen Hubs in ihrer Laubmengensumme das maximale Ladevolumen $\overline{T}$ nicht überschreiten. Im Folgenden wird also davon ausgegangen, dass nur Sammeltouren $K \to h_1 \to h_2 \to \ldots \to h_P \to K$ betrachtet werden, für die gilt:

$$\sum_{p=1,\ldots,P-1} M_s^{**}(h_p) < \overline{T}, \tag{2.24}$$

wobei mit $M_s^{**}(h_p)$ die nach entsprechend vielen Pendeltouren verbleibende Laubrestmenge beim Hub h_p gemeint ist. Hiermit wird sichergestellt, dass bei Weiterfahrt zum nächsten Hub noch Aufladekapazität in der Schubkarre vorhanden ist. Allerdings wird offen gelassen, ob der letzte Laubhaufen komplett aufgeladen werden kann oder nur ein Teil der Laubmenge $M_s^{**}(h_P)$ auf der Schubkarre Platz findet. Dann ergibt sich der Transportaufwand für diese Sammeltour im einfachen Fall, d.h. in Anlehnung an (2.19), zu

$$TA(h_1, \ldots, h_P) = \alpha_T \cdot [d_{K,h_1}^* + d_{h_P,K}^* + \sum_{p=2,\ldots,P} d_{h_{p-1},h_p}^*]. \tag{2.25}$$

Auf weitere Verfeinerungen soll hier verzichtet werden; hierzu siehe [13], S. 79.

Zur Ermittlung von effizienten Transportabfolgen, insbesondere bei den Sammeltouren, empfiehlt sich der Einsatz von Lösungsverfahren für Rundreise- und Tourenplanungsprobleme[28]. Bei den Vergleichstest (Kapitel 6) werden der Einfachheit

[28] Näheres hierzu siehe u.a. in [6].

halber allerdings ausschließlich Pendeltouren (Hubs $\leftrightarrow$ Kompoststelle) benutzt, zumal der Harkprozess im Vordergrund steht.

2.3 Das formale Laubharkmodell

Die in den obigen Abschnitten dieses Kapitels ausführlich hergeleitete mathematische Modellierung eines Laubharkproblems soll hier auf seinen formalen Kern zusammengefasst werden.

Gegeben sei ein kanten- und knotenbewerteter Graph $G = [V, E, c, M]$ mit der Knotenmenge V, der Kantenmenge E, der Kantenbewertungsfunktion[29] $c : E \to \mathbb{R}_{\geq}$ und der Knotenbewertungsfunktion $M : V \to \mathbb{R}_{\geq}$.

Eine Abbildung $s : V \to V$ heißt ***Nachfolgerfunktion*** von G, wenn s die ***Nachbarschaftsbedingung*** erfüllt (d.h. für alle $a \in V$ gilt: $a \neq s(a) \Rightarrow [a, s(a)] \in E$; vgl. (2.9)), und wenn zudem der durch s induzierte Digraph $\vec{G}_s = \langle V, \vec{E}_s \rangle$ mit $\vec{E}_s = \{\langle a, s(a)\rangle \mid a \in V \wedge a \neq s(a)\}$ zyklenfrei ist. Die Menge aller Senken in $\vec{G}_s$ werde mit $Hub(G, s)$ bezeichnet.

Zu jedem $a \in V$ ist bzgl. $\vec{G}_s$ die *Erreichbarkeitsmenge* $R(a)$ definiert, d.h. die Menge aller Knoten $i \in V$, von denen aus der Knoten a über eine Pfeilfolge in $\vec{G}_s$ erreichbar ist; dabei werde $a \in R(a)$ vereinbart. Die Erreichbarkeitsmengen der Senken in $\vec{G}_s$ werden auch ***Cluster*** genannt und mit $C(h)$ für $h \in Hub(G, s)$ bezeichnet.[30] Zudem wird die Abbildung $M_s^* : V \to \mathbb{R}_{\geq}$ durch $M_s^*(a) = \sum_{k \in R(a)} M(k)$ definiert; sie wird die durch s induzierte ***Anhäufungsfunktion*** genannt.

Die Menge aller Nachfolgerfunktionen von G werde mit $\Lambda(G)$ bezeichnet. Durch eine *Kostenfunktion* $K : \Lambda(G) \to \mathbb{R}_{\geq}$ werden jeder Nachfolgerfunktion s von G nichtnegative Kosten zugewiesen.[31] Dann wird das Optimierungsproblem in der Form

$$\min_{s \in \Lambda(G)} K(s) \tag{2.26}$$

[29] Siehe Seite 11.

[30] Im Folgenden wird der Begriff des Clusters nicht nur dazu verwendet, eine spezielle Teilmenge C der Knotenmenge V zu benennen, sondern vereinzelt auch den durch die (nichtleere) Teilmenge $C \subset V$ induzierten Untergraph $G(C)$ von G, wobei C per Konstruktion stets zusammenhängend ist.

[31] Spezielle Kostenfunktionen ergeben sich durch die Aufsummierung der Aufwände für Harken, unproduktive Wege und Transporte; siehe (2.28).

ein ***Laubharkproblem*** genannt. Dabei wird eine kostenminimale Nachfolgerfunktion s von G gesucht.

Mit der Abbildung $\overline{M} : V \to \mathbb{R}_{\geq}$ werde eine ***Kapazitätsfunktion*** eingeführt. Eine Nachfolgerfunktion s heißt ***zulässig*** bzgl. $\overline{M}$, wenn für alle Knoten $a \in V$ gilt: $M_s^*(a) \leq \overline{M}(a)$.[32] Dann wird das Optimierungsproblem in der Form

$$\begin{aligned} &\min_{s \in \Lambda(G)} K(s) \\ &u.d.N. \\ &M_s^*(a) \leq \overline{M}(a) \text{ für alle } a \in V \end{aligned} \tag{2.27}$$

ein ***kapazitiertes Laubharkproblem*** genannt. Dabei wird eine kostenminimale zulässige Nachfolgerfunktion s von G gesucht. Im einfachsten Fall gilt $\overline{M}(a) = \overline{M} \in \mathbb{R}_{>}$ für alle $a \in V$, wobei dann die Nebenbedingungen allein nur für die Senken $h \in Hub(s)$ gefordert werden müssen.

Im Folgenden wird davon ausgegangen, dass ein kapazitiertes Laubharkproblem mit konstanter Laubmengenobergrenze $\overline{M}$ zu Grunde liegt, für das eine effiziente Lösung in Form einer Nachfolgerfunktion s eines gegebenen knoten- und kantenbewerteten Graphen $G = [V, E, c, M]$ gesucht wird. Dabei setze sich die zu minimierende Zielfunktion $K(s)$ aus den additiven Komponenten $HA_\Sigma(s)$ gemäß (2.12), (2.18), $WA_\Sigma(s)$ gemäß (2.15) und $TA_\Sigma(s)$ gemäß (2.23) zusammen:

$$K(s) = HA_\Sigma(s) + WA_\Sigma(s) + TA_\Sigma(s). \tag{2.28}$$

Im Fokus der Betrachtungen steht im Folgenden der Harkprozess und damit die Entwicklung von effizienten Harkstrategien. Dabei wird von den ermittelten Nachfolgerfunktionen, welche die Harkstrategien charakterisieren, die Zulässigkeit durch eine einfache Prüfroutine sichergestellt.

2.4 Vergleich mit ähnlichen Optimierungsproblemen

Das Laubharkproblem (LHP) gehört gemäß der vorgestellten Modellierung in Abschnitt 2.3 zur Klasse der kombinatorischen Optimierungsprobleme[33], zumal die

[32] Vgl. (2.11).

[33] Es sei allerdings angemerkt, dass es sich beim ursprünglichen Laubharkproblem eigentlich um ein kontinuierliches Problem handelt, zumal theoretisch jeder Punkt der Ebene als Hub ausgewählt werden könnte. Die Diskretisierung des Problems geschieht erst durch die Einteilung der Gartenfläche in Felder.

Lösungsmenge endlich ist.[34] Die Lösungsfindung für ein LHP besteht im Wesentlichen aus einer geeigneten Bestimmung von Sammelstellen (Hubs) und einer passenden Zuweisung der Felder zu diesen Hubs (Clusterbildung). Damit ähnelt das LHP in gewisser Weise einem Facility-Location-Problem (FLP) oder auch einem Hub-Location-Problem (HLP).[35]

Ein FLP steht an, wenn anstelle von direkten Belieferungen der Kunden(orte) von einem Zentrallager oder Werk die Kundenbelieferungen (ausschließlich) über Zwischenlager oder Filialen vorgenommen werden sollen. Zum einen sind diese Zwischenlager (Hubs) zu bestimmen, zum anderen ist eine Zuweisung der Kunden(orte) zu den gewählten Hubs vorzunehmen. Bei einem LHP übernimmt die Kompoststelle die Rolle des Zentrallagers, die Felder werden als Kunden und deren Laubmengen als Kundenbedarfe interpretiert. Darüber hinaus stellt sich ein Richtungswechsel ein, indem die „Lieferungen“ nicht vom Zentrallager über die Zwischenlager zu den Kunden(orten), sondern umgekehrt von den Feldern (Kunden) über die Sammelstellen (Hubs) zur Kompoststelle (Zentrallager) erfolgen. Da die Zuweisung eines Harkfeldes zum Hub eindeutig ist, liegt ein FLP vom Single-Allocation-Typ vor.[36] Da zudem die Felder (und somit auch die Hubs) bei LHP in der Regel mit Laubmengenobergrenzen versehen sind, interessieren vergleichsweise die kapazitierten FLP. Ohne Berücksichtigung der Zielfunktion lässt sich ein LHP also zunächst als ein „inverses“ FLP ansehen.

Bei einem HLP werden ebenfalls Sammelstellen (Hubs) als Zwischenstationen für den Transport von Objekten (z.B. Güter, Personen, Daten) zwischen verschiedenen Auslieferungs- und Empfangsorten[37] bestimmt, um dadurch die Gesamttransportkosten zu verringern. Dabei ist speziell die Klasse der diskreten, endogenen HLP vom Single-Allocation-Typ mit kapazitierten Hubs bei nicht festgelegter Hubanzahl von Interesse.[38] Bei der Betrachtung eines LHP als spezielles HLP treten allerdings einige Besonderheiten ein: Es gibt nur einen einzigen Empfangsort (Kompoststelle), während alle anderen Knoten des Netzwerkes ausschließlich Auslieferungsorte sind.[39] Die Kosten für eine „Lieferung“ von einem Nicht-Hub

[34] Der Lösungsraum besteht bei n Knoten theoretisch aus bis zu k^n (nicht notwendigerweise zulässigen) Lösungsmöglichkeiten, wobei der Basiswert k von der gewählten Nachbarschaft der Felder (bzw. Knoten) abhängt; bei den Nachbarschaftsvarianten (siehe Abbildung 4) gilt beispielsweise $k = 5$ für Variante a bzw. $k = 9$ für Variante b.

[35] Die Klasse der Hub-Location-Probleme wird in der Literatur meistens als ein Spezialfall aus der Klasse der Facility-Location-Probleme aufgeführt („*Hub location problem (HLP) is one of the popular topics in FLP, known as hub and spoke problem …*“, [23], Seite 92).

[36] Zur Klassifizierung von Facility-Location-Problemen siehe u.a. [11], [21].

[37] In der Regel ist jeder Ort sowohl Auslieferungs- und Empfangsort, wobei Export- und Importmenge eines Ortes unterschiedlich ausfallen (können).

[38] Zur Klassifikation von Hub-Location-Probleme siehe u.a. [1], [4], [8], [23].

[39] Die Kompoststelle ließe sich dabei auch als „Super-Hub“ betrachten.

zu seinem Hub stehen nicht von vornherein fest, sondern hängen von der Liefertour ab, die sich aus der Lösungsstruktur ergibt. Zwischen den Hubs gibt es keine Lieferungen, sondern nur von den Hubs zum Empfangsort. Die günstigeren Transportstückkosten zwischen den Hubs lassen sich somit auch nicht durch einen Diskontfaktor steuern; vielmehr geschieht eine Kostenvergünstigung durch eine unterschiedliche Kostenverrechnung des Harkprozesses („Transport von Auslieferungsorten zu Hubs“) und des eigentlichen Transportprozesses („Transport von den Hubs zum Empfangsort“).

Zu beiden Vergleichsfällen (FLP, HLP) unterscheidet sich ein LHP demnach in einigen wesentlichen Punkten, die hauptsächlich in der speziell zusammengesetzten Zielfunktion begründet sind. Die zu minimierende Aufwandsfunktion des LHP setzt sich aus drei Teilaufwänden zusammen (Harkaufwand, Transportaufwand, Aufwand für unproduktive Wege), die nicht einzeln, sondern in ihrer Summe minimal ausfallen sollen. Dabei ist der Harkaufwand bei einer LHP-Lösung – wie bereits oben erwähnt – von der Clusterbildung und der Zuweisungsstruktur innerhalb der Cluster abhängig. Zudem weisen Hark- und Transportaufwände ein gegenläufiges Verhalten hinsichtlich der Anzahl der Cluster bzw. Hubs auf (je mehr Cluster, umso weniger Harkaufwand, aber umso mehr Transportaufwand und umgekehrt). Letztlich ist der Aufwand für unproduktive Wege als dritte Kostenkomponente weder in FLP- noch in HLP-Modellen verankert. Somit bilden LHP-Modelle durchaus eine eigene Problemklasse unter den kombinatorischen Optimierungsproblemen.

Eine mathematische Modellierung des LHP mittels Binärvariablen, wie es bei kombinatorischen Problemen und speziell auch bei Zuweisungsproblemen (z.B. auch bei FLP und HLP) durchaus üblich ist, ist zwar in Ansätzen möglich, allerdings führen einige Nebenbedingungen zu Komplikationen; so ist es sehr aufwändig bzw. wenig effizient, die für Harkvorschriften benötigte Zyklenfreiheit mit Hilfe von (Un-)Gleichungen und Binärvariablen zu erzwingen[40] oder damit die Aufwände für unproduktive Wege zu beschreiben. Daher wird hier auf ein derartiges Formalmodell verzichtet.

[40] Hierzu vergleiche man den Aufwand zum Vermeiden von Subzyklen in Modellierungsansätzen für das Traveling-Salesman-Problem; siehe u.a. [6].

3 Deterministische Harkstrategien für das Laubharkproblem

Zur Bestimmung einer effizienten[41] Lösung für das Laubharkproblem gemäß (2.27) und (2.28) wurde eine Vielzahl von Laubharkstrategien entwickelt und untersucht. In diesem Kapitel sollen zunächst ***deterministische Harkstrategien*** vorgestellt werden. Dabei handelt es sich um solche Strategien, deren jeweilige Systematik durch einfache deterministische Regeln festgelegt ist. Dabei basieren einige Strategien auf intuitive Vorgehensweisen. Ein typisches Beispiel dafür ist eine simple *Zickzack-Strategie*, wie sie bereits in Kapitel 2 an Beispielen zur Anwendung kommt.[42]

Einige dieser deterministischen Strategien erweisen sich als besonders effizient, insbesondere in der Rolle als *Eröffnungs-* oder *Konstruktionsverfahren*, also als einfache und schnelle Algorithmen zur Auffindung von zulässigen Lösungen mit suboptimaler, aber möglichst „akzeptabler" Lösungsgüte, wobei die Effizienz am Verhältnis dieser Lösungsgüte zur Rechenzeit gemessen wird.[43]

Bei allen untersuchten Harkstrategien sind grundsätzlich die beiden Schwerpunkte ***Clusterbildung*** und ***Hubbestimmung*** zu beachten. Dabei steht zunächst die Clusterbildung im Vordergrund („Cluster first"), wobei zumeist auch *implizit* eine (vorläufige) Hubbestimmung erfolgt, die allerdings von einer sich anschließenden *expliziten* Hubbestimmung überschrieben werden kann. Bei der Clusterbildung kann eine *sukzessive* oder eine *simultane* Vorgehensweise verfolgt werden. Man beachte, dass in den folgenden Vorgehensbeschreibungen vornehmlich von Knoten statt von Feldern gesprochen wird.

---

[41] Da es sich bei den Verfahren um heuristische Methoden handelt, wird der Begriff „optimal" (bzw. „minimal") tunlichst vermieden.

[42] Hierzu siehe Abbildung 8 und Abbildung 9. Eine ausführlichere Beschreibung dazu erfolgt in Abschnitt 3.1.

[43] Näheres zum Effizienzbegriff siehe im Anhang A.1.

3.1 Sukzessive Clusterbildung

Bei der ***sukzessiven Clusterbildung*** wird zunächst durchgängig *ein aktuelles* Cluster betrachtet, welches durch die Zuweisung von geeigneten Knoten „vervollständigt“ wird, bevor ein neues Cluster in Augenschein genommen wird („1-Open-Type“). Hierbei wird ein noch nicht zugewiesener Knoten als ***Repräsentant*** ausgewählt, welcher das Cluster „eröffnet“. Durch die Wahl dieser Repräsentanten erfolgt eine implizite Hubbestimmung, da diese ausgewählten Knoten die Rolle von vorläufigen Hubs annehmen. Dann werden schrittweise „passende“ Knoten dem aktuellen Cluster hinzugefügt, bis kein solcher Knoten mehr zu finden ist. Ob ein Knoten für die Aufnahme in das aktuelle Cluster „passend“ ist oder nicht, wird anhand der folgenden „Fit-Kriterien“ überprüft:

(1) Der Aufnahmekandidat ist noch keinem Cluster zugewiesen.
(2) Der Aufnahmekandidat muss mit einem im aktuellen Cluster befindlichen Knoten benachbart sein.
(3) Durch die Hinzunahme der Laubmenge des Aufnahmekandidaten zur bisherigen Laubmengensumme im Cluster darf die maximal erlaubte Laubmenge $\overline{M}$ nicht überschritten werden.

Wenn mehrere Knoten diese Fit-Kriterien erfüllen und somit mehrere Aufnahmekandidaten in Frage kommen, muss ein eindeutiges ***Aufnahmekriterium*** über die konkrete Zuweisung entscheiden. Hier eignet sich beispielsweise das eindeutige „Kleinster-Index-Kriterium“, d.h. unter allen Aufnahmekandidaten wird der Knoten mit dem kleinsten Knotenindex gewählt. In der Felddarstellung des Gartens entspricht dieses Kriterium der sogenannten *Nordwest-Ecken-Regel*, wobei unter allen am meisten nördlich liegenden Feldern das am meisten westlich liegende Feld bestimmt wird.

Für die Implementierung dieser Kriterien bietet es sich an, eine ***Kandidatenliste*** aufzustellen und abzuarbeiten. Sobald ein Kandidat dem aktuellen Cluster zugewiesen worden ist[44], werden alle Nachbarknoten des Kandidaten geprüft, ob sie bereits einem Cluster zugewiesen sind oder bereits in der Kandidatenliste aufgenommen wurden. Wenn nicht, werden sie der Kandidatenliste in einer bestimmten Reihenfolge zugefügt. Dabei sind mehrere Reihenfolgekriterien denkbar:

(1) nach aufsteigendem Knotenindex (*Nordwest-Ecken-* oder *Kleinster-Index-Regel*, kurz: NW)
(2) nach absteigender Laubmengengröße (LM_{max})
(3) nach aufsteigender Laubmengengröße (LM_{min})

[44] Dies gilt anfänglich für den Repräsentanten. Der zugewiesene Kandidat wird aus der Kandidatenliste entfernt.

Zudem ist zu unterscheiden, ob es sich um eine ***Schlangenliste*** (FIFO $\equiv$ First In - First Out) oder eine ***Stapelliste*** (LIFO $\equiv$ Last In - First Out) handelt. Mit der Verwendung einer Schlangen- bzw. Stapelliste geht eine ***Breiten-*** bzw. ***Tiefensuche*** (BS bzw. TS) einher.[45] Je nach Listentyp wird der nächste Kandidat der Liste entnommen und auf das dritte Fit-Kriterium hin geprüft.[46] Sobald die Kandidatenliste leer ist, aber noch nicht alle Knoten eine Clusterzuweisung erhalten haben, wird ein neues Cluster eröffnet.

Schließlich spielt bei der Clusterbildung[47] auch die ***Repräsentantenwahl*** eine wesentliche Rolle. Hier bieten sich beispielsweise folgende Möglichkeiten an: Gewählt wird unter den noch nicht zugewiesenen Knoten

(1) der „nordwestlichste“ Knoten (NW).
(2) der Knoten mit der größten Laubmenge (LM_{max}).
(3) der Knoten mit der geringsten Entfernung zum Kompostknoten (KP_{min}).

Die Eindeutigkeit wird ggf. durch die Kleinster-Index-Regel als Sekundärkriterium hergestellt.

Zu den heuristischen Verfahren mit sukzessiver Clusterbildung gehört auch die intuitive *Zickzack-Strategie*, die stringent den Aufbau *Cluster first - Hub second* einhält.[48] Entsprechend der unteren Skizze wird vom Feld mit dem kleinsten Index (Nordwestecke) in waagerechter Richtung geharkt, beginnend von Westen nach Osten bzw. von links nach rechts, bis die jeweilige Gartengrenze erreicht ist. Danach wird senkrecht in das untere Feld und von dort in die umgekehrte waagerechte Richtung weiter geharkt (falls möglich), also abwechselnd von links nach recht bzw. von rechts nach links. Dieser Harkvorgang wird allerdings unterbrochen, wenn die kumulierte Harklaubmenge die vorgegebene Laubmengenobergrenze $\overline{M}$ überschreitet oder ein blockiertes Feld erreicht wird. Im ersten Fall wird das letzte Feld, welches die kumulierte Laubmenge noch aufnehmen kann, zum Repräsentanten erklärt und es wird beim nächsten harkbaren Feld in jeweilige Harkrichtung mit dem Laubharken wieder neu begonnen. Im zweiten Fall wird auch hier das letzte Feld zum Repräsentanten erklärt und das nächste zulässige Feld in Zickzack-Richtung gesucht, um dort mit dem Harken fortzusetzen (vgl. folgende Skizze).

[45] Dabei bilden sich Cluster in „Kugelform“ bzw. „Stern-“ oder „Krakenform“.
[46] Man beachte, dass die Überprüfung der ersten beiden Fit-Kriterien bereits bei der Aufnahme in die Kandidatenliste erfolgt ist.
[47] Dies betrifft sowohl die sukzessive als auch die simultane Clusterbildung.
[48] Durch die Bestimmung von vorläufigen Hubs (Repräsentanten) liegt bei den obigen Vorgehensweisen eher der Aufbau *Hub first - Cluster second* vor, auch wenn es sich nur um vorläufige Hubs handelt.

$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	h_1	$\rightarrow$	$\downarrow$
$\downarrow$	$\leftarrow$	$\leftarrow$		h_2	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$
$\rightarrow$	$\rightarrow$	$\rightarrow$	h_3			$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\downarrow$
h_5	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	h_4	$\leftarrow$	$\leftarrow$

Sukzessive Clusterbildung via BS_{NW}, BS_{max}, BS_{komp}

Gegeben: Knotenmenge $V \neq \emptyset$, zugehörige Adjazenzmatrix, Knotenbewertungsvektor M, mit Nullen initialisierter Vektor s sowie die Entfernungen $d_K(a)$ aller Knoten $a \in V$ zur Kompoststelle K.[a]

Schritt 1:

Fall * $\equiv NW$: Wähle Knoten $a \in V$ mit kleinstem Index .[b]

Fall * $\equiv max$: Wähle Knoten $a \in V$ mit $M(a) = \max\{M(i) \mid i \in V\}$.[c]

Fall * $\equiv komp$: Wähle Knoten $a \in V$ mit $d_K(a) = \min\{d_K(i) \mid i \in V\}$.[d]

Setze $h := a; V := V \setminus \{h\}, M_s^*(h) := M(h); s(h) := h; K := \emptyset$.[e]

Schritt 2:

Für alle $i \in (N(a) \cap V) \setminus K$, führe aus:

Falls $M_s^*(h) + M(i) \leq \overline{M}$, führe aus:

Setze $K := K \cup \{i\}; V := V \setminus \{i\}; s(i) := a; M_s^*(h) := M_s^*(h) + M(i)$.

Schritt 3:

Falls $K \neq \emptyset$, führe aus:

Wähle $a \in K$ mit kleinstem Index.[f]

Gehe zu Schritt 2.

Falls $V \neq \emptyset$, gehe zu Schritt 1.

Terminiere!

[a] Anhand der Adjazenzmatrix lassen sich die Nachbarmengen $N(i), i \in V$, bestimmen. Für alle Knotenbewertungen gilt: $M(i) \leq \overline{M}, i \in V$. Der Vektor s dient als Nachfolgerfunktion ($s \equiv$ successor).

[b] Da die Knotenmenge V auf kanonische Weise in geordneter Form vorliegt, ist dies identisch mit der Wahl des ersten Knotens in der Liste.

[c] Falls anhand des Maximum-Kriteriums keine eindeutige Knotenbestimmung möglich ist, wird unter den alternativen Knoten derjenige mit dem kleinsten Index gewählt.

[d] Falls anhand des Minimum-Kriteriums keine eindeutige Knotenbestimmung möglich ist, wird unter den alternativen Knoten derjenige mit dem kleinsten Index gewählt.

[e] Die Kandidatenliste K ist ein Schlangenspeicher (FIFO).

[f] Die Wahl des Knotens mit kleinstem Index entspricht der FIFO-Regel.

3.2 Simultane Clusterbildung

Bei der ***simultanen Clusterbildung*** bleiben alle zwischenzeitlich entstandenen Cluster durchgehend zur Aufnahme von weiteren Knoten offen („More-Open-Type“). Dabei wird jeder Knoten genau einmal erfasst und behandelt, wonach sich der jeweilige Erfassungszustand ändert („noch nicht erfasst“ $\rightarrow$ „erfasst“). In

jedem Schritt wird unter den noch nicht erfassten Knoten[49] ein ***Zuweisungskandidat*** bestimmt. Als Auswahlkriterium sind viele Möglichkeiten denkbar. Beispielsweise wird der Knoten mit der geringsten Laubmenge unter den noch nicht erfassten Knoten ausgewählt (Sekundärkriterium: Kleinster Index). Dadurch findet eine Knotenabarbeitung nach aufsteigenden Laubmengengrößen statt (LM_{min}). Anschließend wird ein Nachbarknoten des Zuweisungskandidaten als ***Kontaktkandidat*** ausgewählt. Hierfür sind ebenfalls verschiedene Auswahlkriterien möglich. Beispielsweise wird der Nachbarknoten mit der größten Laubmenge als Kontaktkandidat ausgewählt (Sekundärkriterium: Kleinster Index); dabei wird geprüft, ob die Laubmengenobergrenze eingehalten wird, wobei eine fallspezifische Zulässigkeitsprüfung zum Tragen kommt; ansonsten wird der Nachbar mit der nächstkleineren Laubmenge geprüft. Wird ein passender Nachbarknoten gefunden, sind für das ausgewählte Knotenpaar (Zuweisungskandidat & Kontaktkandidat) vier Fälle hinsichtlich ihrer Clusterzugehörigkeit möglich:

(1) Beide Knoten sind noch keinem Cluster zugewiesen.
Beide Knoten werden einem neuen Cluster zugewiesen. Entsprechende Zulässigkeitsprüfung: Die beiden Laubmengen zusammen überschreiten nicht die Laubmengenobergrenze.

(2) Nur der Kontaktkandidat ist einem Cluster zugewiesen.
Der Zuweisungskandidat wird dem Cluster des Kontaktkandidaten zugewiesen. Entsprechende Zulässigkeitsprüfung: Die Hinzunahme der Laubmenge des Zuweisungskandidaten zur kumulierten Laubmenge des Clusters vom Kontaktkandidaten überschreitet nicht die Laubmengenobergrenze.

(3) Nur der Zuweisungskandidat ist einem Cluster zugewiesen.
Der Kontaktkandidat wird dem Cluster des Zuweisungskandidaten zugewiesen. Entsprechende Zulässigkeitsprüfung: Die Hinzunahme der Laubmenge des Kontaktkandidaten zur kumulierten Laubmenge des Clusters vom Zuweisungskandidat überschreitet nicht die Laubmengenobergrenze.

(4) Beide Knoten sind bereits einem Cluster zugewiesen.
Im trivialen Fall, dass beide Knoten zum selben Cluster gehören, findet keine weitere Zuweisungshandlung statt (ebenso keine Zulässigkeitsprüfung). Andernfalls wird geprüft, ob die kumulierten Laubmengen der beiden Cluster in der Summe die Laubmengenobergrenze nicht überschreiten. Dann werden die beiden Cluster vereint; in praxi werden die Knoten aus dem Cluster mit dem kleineren Clusterindex dem Cluster mit dem größeren Clusterindex zugewiesen.[50]

[49] Dabei wird (noch) nicht unterschieden, ob der Knoten bereits einem Cluster zugewiesen ist oder nicht.

[50] Die dadurch entstehenden leeren Cluster werden in einem späteren Schritt eliminiert, indem die Clusterindizes entsprechend verringert werden.

Simultane Clusterbildung

Gegeben: Knotenmenge $V \neq \emptyset$, Nachbarmengen $N(i), i \in V$, Knotenbewertungsvektor M .[a]

Schritt 1:

Für alle $a \in V$ setze: $C(a) := \emptyset, M^*(C(a)) := 0$.[b]

Schritt 2:

Wähle Knoten $a \in V$ mit $M(a) = \min\{M(i) \mid i \in V\}$.[c]

Setze $V := V \setminus \{a\}$.

Falls $C(a) = \emptyset$, setze $V_a := M(a)$;

andernfalls setze $V_a := M^*(C(a))$.

Setze $K := \emptyset$.

Für alle $b \in N(a) \setminus C(a)$ führe aus:

Falls $C(b) = \emptyset$, setze $V_b := M(b)$;

andernfalls setze $V_b := M^*(C(b))$.

Falls $V_a + V_b \leq \overline{M}$, setze $K := K \cup \{b\}$.

Falls $K \neq \emptyset$, führe aus:

Wähle Knoten $b \in K$ mit $M(b) = \max\{M(i) \mid i \in K\}$.[d]

Falls $C(a) = \emptyset$, führe aus:

Falls $C(b) = \emptyset$, setze:

$$C(a) := \{a, b\}, C(b) := C(a), M^*(C(a)) := M(a) + M(b);$$

andernfalls setze:

$$M^*_{alt} := M^*(C(b)), C(b) := C(b) \cup \{a\}, C(a) := C(b),$$
$$M^*(C(b)) := M^*_{alt} + M(a);$$

andernfalls ($C(a) \neq \emptyset$), führe aus:

Falls $C(b) = \emptyset$, setze:

$$M^*_{alt} := M^*(C(a)), C(a) := C(a) \cup \{b\}, C(b) := C(a),$$
$$M^*(C(a)) := M^*_{alt} + M(b);$$

andernfalls vereinige die Cluster $C(a)$ und $C(b)$.[e]

Andernfalls ($K = \emptyset$) führe aus:

Falls $C(a) = \emptyset$, setze $C(a) := \{a\}, M^*(C(a)) := M(a)$.

Falls $V \neq \emptyset$, gehe zu Schritt 2.

---

[a] Für alle Knotenbewertungen gilt: $M(i) \leq \overline{M}, i \in V$.

[b] $C(a)$ ist das Cluster, dem der Knoten a zugewiesen ist, wobei $\emptyset$ bedeutet, dass der Knoten a noch keinem Cluster zugewiesen ist. M^* gibt die aktuell kumulierte Laubmenge des jeweiligen Clusters an.

[c] Falls anhand des Minimum-Kriteriums keine eindeutige Knotenbestimmung möglich ist, wird unter den alternativen Knoten derjenige mit dem kleinsten Index gewählt.

[d] Falls anhand des Maximum-Kriteriums keine eindeutige Knotenbestimmung möglich ist, wird unter den alternativen Knoten derjenige mit dem kleinsten Index gewählt.

[e] Entsprechend sind $C(a)$ bzw. $C(b)$ sowie $M^*(C(a))$ bzw. $M^*(C(b))$ zu aktualisieren.

Falls in einem Erfassungsschritt für den aktuellen Zuweisungsknoten kein passender Kontaktkandidat gefunden werden kann, allerdings bereits eine Clusterzuweisung vorliegt, wird keine Zuweisungshandlung vorgenommen. Andernfalls wird ein neues Cluster eröffnet, bei dem der Zuweisungsknoten als Repräsentant fungiert. Das Verfahren endet, wenn jeder Knoten erfasst worden ist.

Die algorithmische Beschreibung der simultanen Clusterbildung (siehe Seite 36) in der oben ausgeführten Art und Weise lässt sich durch andere Aufnahmekriterien variieren.

Zur Klasse der Verfahren mit simultaner Clusterbildung gehören auch die folgenden *modifizierten Fit-Strategien*, welche die bekannten Fit-Strategien zur Lösung von eindimensionalen Bin-Packing-Problemen nachahmen (*First Fit, Best Fit*).[51] Hierbei werden die Cluster als Bins, die Knoten als Items und die zugehörigen Laubmengen als Längen aufgefasst. Die Bins haben die Laubmengenobergrenze als einheitliche Länge.[52]

Die *modifizierte First-Fit-Strategie* (FF_{mod}) hat folgenden Ablauf:
Jeder Knoten wird in Knotenindexreihenfolge einem Cluster zugewiesen. Dabei wird zunächst jedes bereits vorhandene Cluster in Clusterindexreihenfolge überprüft, ob dafür genügend Gap[53] im Cluster vorhanden ist. Wenn ja, wird geprüft, ob der aktuelle Knoten zu einem Clusterelement benachbart ist.[54] Wenn beide Prüfkriterien erfüllt sind, wird der aktuelle Knoten dem ersten Cluster, für das es passt, zugewiesen („first fit“). Wenn kein Cluster passend ist, wird ein neues Cluster geöffnet. Bei der *First-Fit-Decrease-Variante* (FFD_{mod}) werden die Knoten nach absteigender Laubmengengröße aufgenommen. Ansonsten wird wie beim First-Fit verfahren. Da bei der Laubharkproblematik von Anfang an alle Knoten und zugehörigen Laubmengen bekannt sind, liegt eine *Offline-Situation* vor, die es rechtfertigt, auch „Decrease-Varianten“ als Lösungsstrategien einzubinden.

Die *modifizierte Best-Fit-Decrease-Strategie* (BFD_{mod}) hat folgenden Ablauf:
Jede Clustereröffnung erfolgt mit dem aktuell laubmengengrößten Knoten[55] als Repräsentant (vorläufiger Hub).[56] Es wird unter den noch nicht zugewiesenen Knoten derjenige Knoten gesucht, dessen Aufnahme in ein bestehendes Cluster

[51] Näheres hierzu siehe u.a. [12].
[52] Am Rande sei bemerkt, dass diese Vorgehensweise nur für den Fall, dass alle Felder dieselbe Laubmengenobergrenze besitzen, konzipiert ist.
[53] Gemeint ist die Differenz zwischen maximaler und kumulierter Laubmenge des Clusters.
[54] In diesem zweiten Prüfkriterium steckt die wesentliche Modifikation gegenüber der First-Fit-Strategie für Bin-Packing-Probleme.
[55] D.h. unter allen noch nicht zugewiesenen Knoten.
[56] Demnach handelt es sich um eine „Decrease-Variante“.

den geringstmöglichen Harkaufwand nach sich zieht; dies wird gemessen an der Entfernung zum Repräsentanten. Mit anderen Worten: Jeder Knoten wird in absteigender Laubmengengröße einem Cluster zugewiesen. Dabei wird für einen „Aufnahmekandidaten“ der Reihe nach jedes offene Cluster überprüft, ob dafür genügend Gap im Cluster vorhanden ist. Wenn ja, wird geprüft, ob der aktuelle Knoten zu einem Clusterelement benachbart ist. Falls es mehr als einen solchen „Verbindungsknoten“ gibt, wird derjenige bestimmt, der die kürzeste Entfernung zum Repräsentanten hat. Als Vergleichsgröße für die „beste“ Clusterzuweisung gilt der minimale zusätzliche Harkaufwand. Insgesamt wird also das beste Tripel (Aufnahmekandidat, Clusterkandidat, Verbindungsknoten) bestimmt.

Weitere Fit-Strategien sind Next-Fit-, Worst-Fit- und Fold-Varianten [12]. Die modifizierten Fit-Strategien sind zwar getestet worden, zeigten aber keine lohnenswerte Effizienz.

3.3 Explizite Hubbestimmung

Bei den beschriebenen Clusterbildungen wird auf kanonische Weise bereits eine ***implizite Hubbestimmung*** mitgeliefert, indem die jeweils erwähnten Repräsentanten der Cluster die (vorläufigen) Hubs bilden.[57] Dabei kann eine vorliegende Harkvorschrift ungünstige Harkrichtungen aufweisen, die sich ggf. dadurch verbessern lassen, dass die einzelnen Harkrichtungen in den einzeln Clustern auf den jeweiligen Hubknoten ausgerichtet wird (über kürzeste Entfernungen der einzelnen Knoten eines Clusters zum zugehörigen Hubknoten). Bei dieser ***Hubausrichtung*** bleiben die vorläufigen Hubs als solche unverändert.

Vor Hubausrichtung:

$\rightarrow$	$\rightarrow$	Hub
$\uparrow$	$\leftarrow$	$\leftarrow$

Nach Hubausrichtung:

$\rightarrow$	$\rightarrow$	Hub
$\nearrow$	$\nearrow$	$\uparrow$

Darüber hinaus bietet es sich aber auch an, zu den gebildeten Clustern explizite Hubs zu bestimmen, wodurch die vorläufigen Hubs (Repräsentanten) „überschrieben“ werden. Diese ***explizite Hubbestimmung*** kann sich nach der Lage oder den vorliegenden Laubmengen der Knoten richten.

Lage-orientierte Kriterien:

- Ein einfaches lage-orientiertes Kriterium ist die „Nordwestecken-Regel“ (NW).

[57] Darüber hinaus wird auch bereits eine Harkvorschrift mitgeliefert, zumal die Aufnahme neuer Knoten zu einem Cluster über bestimmte Nachbarknoten erfolgt, die dann als Nachfolger des Aufnahmeknotens gelten.

- Ein weiteres lage-orientiertes Kriterium ist die Ausrichtung nach der Entfernung zum Kompostknoten, wodurch a priori der Transportaufwand begünstigt werden soll. Hierbei wird der Knoten zum (endgültigen) Hub bestimmt, welcher die geringste Entfernung zum Kompostknoten aufweist (unter allen Knoten des jeweiligen Clusters). Dieses Auswahlkriterium (KP_{min}) ist im Allgemeinen nicht eindeutig und benötigt somit ein zusätzliches Sekundärkriterium.

Laubmengen-orientierte Kriterien:
Die Laubmengen-orientierten Kriterien sollen a priori den Harkaufwand begünstigen.

- Im einfachen Fall wird der Knoten zum Hub bestimmt, welcher die maximale Laubmenge aufweist (unter allen Knoten des jeweiligen Clusters). Auch dieses Auswahlkriterium (LM_{max}) ist im Allgemeinen nicht eindeutig und benötigt somit ein zusätzliches Sekundärkriterium.
- Mit etwas mehr Aufwand wird in jedem Cluster der Median bestimmt und zum Hub erklärt. Hierbei handelt es sich um denjenigen Knoten, dessen gewichtete Entfernungssumme zu allen anderen Knoten des jeweiligen Clusters minimal ausfällt, wobei als Gewichte die Laubmengen der Knoten dienen. Auch dieses Auswahlkriterium (LM_{med}) ist im Allgemeinen nicht eindeutig und benötigt somit ein zusätzliches Sekundärkriterium.

Diese Hubbestimmungsmethoden lassen sich folgendermaßen zusammenfassen:

Explizite Hubbestimmung via KP_{min}, LM_{max}, LM_{med}

Gegeben: Clustermenge $C \neq \emptyset$, Knotenbewertungsvektor M, Entfernungsmatrix D, Nachfolgerfunktion s.[a]

Bestimme den Hubknoten $h \in C$ mit[b]

KP_{min}: $d_K(h) = \min\{d_K(i) \mid i \in C\}$.

LM_{max}: $M(h) = \max\{M(i) \mid i \in C\}$

LM_{med}: $\sigma(h) = \min\{\sigma(i) \mid i \in C\}$ mit $\sigma(a) = \sum_{i \in C} \big(M(i) \cdot D(i,a)\big)$ für $a \in C$.

[a] Durch eine zuvor ermittelte Nachfolgerfunktion s ergeben sich auf kanonische Weise die einzelnen Cluster, d.h. jeder Knoten gehört zu genau einem Cluster $C \subset V$.

[b] Falls anhand der Kriterien kein eindeutiger Hubknoten $h \in C$ bestimmt werden kann, wird unter den alternativen Knoten derjenige mit dem kleinsten Index gewählt.

Der expliziten Hubbestimmung schließt sich dann eine entsprechende Hubausrichtung an.

3.4 Verbesserungsmöglichkeiten durch Variation der Laubmengenobergrenze

Die vorgestellten Heuristiken sind in ihren jeweils vorgestellten Grundformen systematisch auf die vorgegebene maximale Laubmenge $\overline{M}$ ausgerichtet, indem ein Hub möglichst bis zu dieser Grenze angehäuft wird, sodass die Laubmengensumme in einem Cluster C möglichst groß ausfällt: $\sum_{a \in C} M(a) \rightarrow \overline{M}$. Dieses Prinzip kann sich allerdings im Hinblick auf die Hauptzielsetzung (Minimierung des Gesamtaufwandes) als kontraproduktiv erweisen, beispielsweise dann, wenn der Transportaufwand einen signifikant geringeren Beitrag zum Gesamtaufwand als der Harkaufwand liefert ($\alpha_T \ll \alpha_H$) oder das maximale Ladevolumen des Transportmittels ($\overline{T}$) kleiner als die maximale Laubmenge pro Feld ausfällt ($\overline{T} < \overline{M}$). Daher empfiehlt es sich, in einer Voruntersuchung den Parameter $\overline{M}$ zu variieren, zumal es sich gezeigt hat, dass die Lösungsgüte der intuitiven Heuristiken signifikant von dieser Parametervoreinstellung abhängen kann, sodass die Ermittlung einer „besten“ Laubmengenobergrenze $\widetilde{M}$ durchaus lohnenswert sein kann. Da die voreingestellte maximale Laubmenge $\overline{M}$ sinngemäß nicht überschritten werden kann bzw. darf, ist folgerichtig nach einer besseren kleineren Laubmengenobergrenze zu suchen ($\widetilde{M} \leq \overline{M}$).

Da der Rechenaufwand für eine möglichst genaue Bestimmung einer empfohlenen Laubmengenobergrenze $\widetilde{M}$ je nach Größe des Suchintervalls und der Schrittweite sehr rechenintensiv sein kann, zumal die Heuristiken entsprechend oft durchlaufen werden müssen, wird als Grundeinstellung empfohlen, den Parameter $\widetilde{M}$ mit dem Parameter $\overline{T}$ im Falle $\overline{T} < \overline{M}$ gleichzusetzen, da diese logisch schlüssige Setzung anhand von diesbezüglich erfolgten Voruntersuchungen grundsätzlich bestätigt wird. Um also die deterministischen Heuristiken im Vergleich zu bionischen Lösungsmethoden (Kapitel 4) „konkurrenzfähiger“ auszustatten, wird von einer Grundeinstellung

$$\widetilde{M} := \begin{cases} \overline{T}, \text{ falls } \overline{T} < \overline{M}, \\ \overline{M}, \text{ sonst.} \end{cases} \tag{3.1}$$

für die deterministischen Verfahren ausgegangen. Allerdings empfiehlt sich die Einbeziehung einer variablen Laubmengenobergrenze $\widetilde{M}$ mit ganzzahliger Schrittweite zu, d.h. die Implementierung einer Subroutine zur Ermittlung der jeweils am besten geeigneten Laubmengenobergrenze.[58]

[58] Diese „beste“ Laubmengenobergrenze kann für jede Strategie verschieden ausfallen. Hierzu siehe auch die Ergebnisse einer Beispielrechnung im Anhang A.2.

3.5 Anmerkungen zu deterministischen Verbesserungsverfahren

Die vorgestellten *deterministischen Heuristiken* lassen sich - wie bereits erwähnt - den sog. Eröffnungs- oder Konstruktionsverfahren zuordnen. Auch wenn durch eine explizite Hubbestimmung bereits ein Verbesserungsschritt vorgesehen ist, so bleibt die Clusterbildung davon zunächst unberührt. Daher könnten sich „echte“ *Verbesserungsverfahren* anschließen. Beispielsweise könnten gezielt ausgewählte Knoten unter den Clustern ausgetauscht werden oder Teile von Clustern mit anderen Teilen von Clustern zu neuen Clustern vereinigt werden. Hier sind viele Möglichkeiten von Clusterumbildungen denkbar. Auch könnte versucht werden, die Harkreihenfolge innerhalb der Cluster gezielt zu ändern. Dabei bliebe allerdings stets zu prüfen, ob nach einer Clusterumbildung bzw. einer Nachfolgeänderung wieder eine gültige Harkvorschrift gegeben ist, von einer Lösungsverbesserung ganz abgesehen. Einige Vorschläge für Verbesserungsansätze findet man in [14], wobei u.a. modifizierte Verfahren für Tourenplanungsprobleme eingesetzt werden (vgl. [6], [16], [3]). Diese deterministischen Verfahren kennzeichnen sich durch ein recht „starres“ Veränderungskonzept, indem sie „nichts dem Zufall überlassen“. Um aber auch zufällige Veränderungen einzubeziehen und damit alternative Wege zur Lösungsverbesserung zu suchen, werden bionische Lösungsansätze ins Spiel gebracht, welche im folgenden Kapitel im Allgemeinen und zwei ihrer Ausprägungen (genetischer und Bienen-Algorithmus) im Besonderen vorgestellt werden.

Im Testprogramm (Kapitel 5) sind neben der einfachen Zickzack-Strategie auch eine Vielzahl von Varianten der sukzessiven und der simultanen Clusterbildung sowie der expliziten Hubbestimmung implementiert. Für einen guten Vergleich der ausgewählten deterministischen Verfahren mit den bionischen Lösungsansätzen hinsichtlich der bikriteriellen Betrachtung von Lösungsgüte und Rechenzeit kann zusätzlich die oben beschriebene Suchroutine zur Ermittlung der jeweils besten Laubmengenobergrenze $\widetilde{M}$ einbezogen werden.

4 Bionische Strategien

Nicht alle Algorithmen zur Optimierung sind strikt deterministisch. Die sogenannten *bionischen Methoden der Optimierung* sind stochastische Metaheuristiken, die in Anlehnung an unterschiedlichste Naturphänomene entwickelt wurden.

Unter bionischen Methoden versteht man Lösungsansätze und –verfahren aus dem Gebiet der Bionik, ein aus Biologie und Technik zusammengesetztes Kofferwort. „Die Bionik (auch Biomimikry, Biomimetik oder Biomimese) beschäftigt sich mit dem Übertragen von Phänomenen der Natur auf die Technik. […] Der Bionik liegt die Annahme zugrunde, dass die belebte Natur durch evolutionäre Prozesse optimierte Strukturen und Prozesse entwickelt, von denen der Mensch lernen kann.“ (aus Wikipedia zum Schlüsselwort „Bionik“).[59]

Zur Lösung des Laubharkproblems wurden ein genetischer Algorithmus (vgl. Abschnitt 4.2), welcher die Erkenntnisse der Evolutionstheorie nutzt, sowie ein schwarmbasierter Bienenalgorithmus problemspezifisch entwickelt (vgl. Abschnitt 4.3).

4.1 Allgemeine Vorgehensweise

Die gängigsten bionischen Methoden folgen einem generellen Grundmuster, welches aus einer Initialisierung der Startlösungen (1.) und einer Wiederholung von Selektion (2.) und Erzeugung (3.) neuer Lösungsanwärter besteht.

1. **Startlösungen:**

   Erzeugung mehrerer Startlösungen (stochastisch/deterministisch). Einzelne Startlösungen können beispielsweise durch Heuristiken aus Kapitel 3 generiert werden (deterministisch). In den meisten Fällen ist es jedoch von Vorteil, einige Startlösungen zufällig zu generieren (stochastisch), denn je besser die Startlösungen im Lösungsraum verteilt sind, umso besser kann die Nachbarschaftssuche in Schritt 3 diesen Raum durchsuchen. Hierbei ist darauf zu achten, dass eventuelle Nebenbedingungen von zufällig erzeugten Startlösungen nicht verletzt werden.

[59] Einen Überblick über die Vielfalt der natur-inspirierten Methoden aus der Bionik gibt die Liste der Metaheuristiken unter https://en.wikipedia.org/wiki/Table_of_metaheuristics.

2. **Selektion:**
   Bewertung der Güte jedes Lösungsanwärters in Anbetracht der Problemstellung (deterministisch) und anschließendes Filtern der Lösungsanwärter nach *guten* Lösungen (stochastisch/deterministisch). Je strikter die Auswahl, umso schnellere Konvergenz wird erzwungen. Dies hat eine positive Auswirkung auf die Laufzeit und erlaubt eine pedantischere Untersuchung der sehr guten Lösungen, kann jedoch zu vorzeitiger Konvergenz gegen lokale Optima führen.
3. **Neue Lösungsanwärter:**
   Aus den in 2. auserwählten Anwärtern werden anschließend neue Lösungsanwärter generiert. Dies geschieht im Allgemeinen über die Nachbarschaftsdefinition, jedoch verwenden manche Verfahren auch andere Methoden, wie beispielsweise die Kreuzung bei genetischen Algorithmen (vgl. Abschnitt 4.2.2). Zudem ist es von Vorteil, in jeder Iteration auch neue zufällige Startlösungen zu erzeugen (stochastisch), um die großflächige Suche im Lösungsraum nicht zu vernachlässigen.

Als gängige Abbruchbedingung für Algorithmen dieser Art betrachtet man die Stagnation der besten Lösung. Verbessert sich diese über viele Iterationen nicht mehr, so bricht der Algorithmus ab. Bei kontinuierlichen Problemen kann es dazu kommen, dass es in sehr naher Umgebung zum Optimum immer wieder zu sehr geringen Verbesserungen pro Iteration kommt. Daher sollte zusätzlich eine generelle Obergrenze an Iterationsschritten festgelegt werden. In diesem Falle würde sich eine genauere Untersuchung auf diese Art nicht weiter auszahlen.

Aufgrund der stochastischen Natur dieser Algorithmen sind sie, unter der Voraussetzung angebrachter Kodierung des Problems, in der Lage, das globale Optimum zu finden. Gängige Heuristiken gehen von gewissen Grundstrukturen aus und nutzen diese. Der Vorteil bionischer Methoden steckt in ihrer Flexibilität. Durch die zufällige und unvoreingenommene Suche im Lösungsraum kann vorzeitige Konvergenz zu lokalen Optima verhindert werden, eine Falle, in die viele Heuristiken laufen. Es ist besonders wichtig sicherzustellen, dass die Bewertung der Güte das betrachtete Problem gut widerspiegelt und dass die Nachbarschaft der Lösungsanwärter sinnvoll definiert ist.

4.1.1 Lösungsanwärter

Lösungsanwärter beim Laubharkproblem sind die Nachfolgerfunktionen (siehe Seite 18). Bei der Erzeugung von Startlösungen (1.) und Bestimmung neuer Lösungsanwärter durch die Manipulation bestehender Lösungen (3.) sind beim Laubharkproblem zwei weitere Bedingungen zu berücksichtigen:

Zyklenfreiheit:
Die entstehende Harkvorschrift darf keine Zyklen enthalten.

Maximale Laubmenge:
Keines der beharkten Felder darf zu irgendeinem Zeitpunkt die maximale Laubmenge überschreiten.

4.1.2 Nachbarschaft

Nachbarschaft in Bezug auf die Kodierung des Problems bedeutet eine geringfügige (nicht zwingend minimale) Veränderung einer Lösung, so dass die Haupteigenschaften erhalten bleiben. Hier gibt es, bezogen auf das Problem der Laubharkoptimierung, verschiedene Ansätze. Für die Implementierung der Nachbarschaft einer Harkvorschrift wird folgende Vorgehensweise (**Nachfolgertausch**) umgesetzt:

Es wird ein Knoten a zufällig bestimmt, dessen Nachfolger $s(a)$ geändert werden soll. Es sei h_1 der Hub des aktuellen Clusters von a und h_2 der Hub des Clusters von a nach dem Nachfolgertausch. Es müssen zwei Bedingungen eingehalten werden. Zum einen kann eine Änderung der Harkvorschrift zur Überschreitung der maximal erlaubten Laubmenge in h_2 führen (falls $h_1 \neq h_2$). Des Weiteren muss die Zyklenfreiheit gewährt bleiben (falls $h_1 = h_2$). In Abbildung 13 ist ein Nachfolgertausch zu sehen, der die Zyklenfreiheit nicht verletzt. Es gilt nach dem Tausch:

$$\exists h_2 \text{ mit } a \in R(h_2), \tag{4.1}$$

d.h. a ist Element der Erreichbarkeitsmenge von h_2 (vgl. Seite 19).

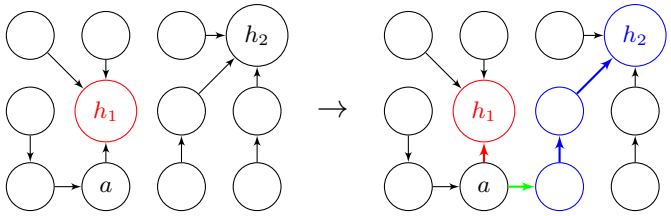


Abbildung 13: *Zulässiger Nachfolgertausch von a (grüner Pfeil).*

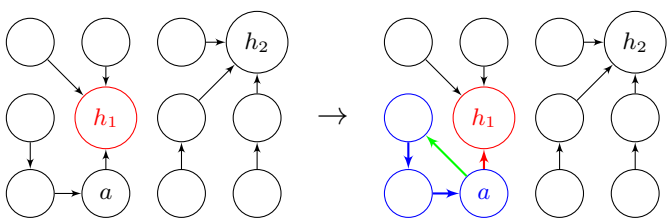


Abbildung 14: *Unzulässiger Nachfolgertausch von a (grüner Pfeil).*

Um sicherzugehen, dass die Kapazitätsgrenze nicht überschritten wird, muss zusätzlich

$$h_1 = h_2 \vee M_s^*(a) + M_s^*(h_2) \leq \overline{M} \tag{4.2}$$

gelten, wobei M_s^* die durch die aktuelle Nachfolgerfunktion s induzierte Anhäufungsfunktion ist (vgl. Seite 19). Beide Bedingungen sind trivialerweise erfüllt, falls $h_1 = h_2$. Dahingegen ist in Abbildung 14 ein unzulässiger Nachfolgertausch zu sehen. Die Zyklenfreiheit ist verletzt.

Ein Spezialfall beim Nachfolgertausch ergibt sich, falls der Knoten a bereits der Hub h_1 des zugehörigen Clusters ist. Durch den Nachfolgertausch wird hierbei eine **Clustervereinigung** durchgeführt, falls keine Verletzung der Zulässigkeit hervorgerufen wird. Dies ist exemplarisch in Abbildung 15 dargestellt.

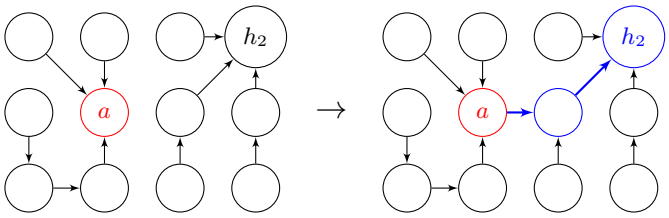


Abbildung 15: *Exemplarische Darstellung einer Clustervereinigung.*

Ein weiterer Spezialfall tritt auf, falls beim Nachfolgertausch der Knoten a auf sich selbst geharkt wird und dadurch zu einem zusätzlichen Hub wird. Dieser Prozess wird als **Clustertrennung** bezeichnet und ist exemplarisch in Abbildung 16 dargestellt.

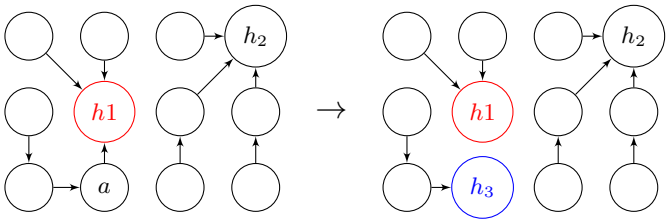


Abbildung 16: *Exemplarische Darstellung der Clustertrennung.*

Bei diesem Fall ist eine Überprüfung der maximal zulässigen Laubmenge in einem Cluster offensichtlich nicht notwendig, so dass dies bei der Implementierung berücksichtigt wurde.

Bei der Beschreibung des Nachfolgertausches werden die bisherigen Hubs bzw. der neu erzeugte Hub für die dazu generierte Harkvorschrift übernommen. Allerdings kann bei Bedarf eine explizite Hubbestimmung für die neuen Cluster nach Abschnitt 3.3 vorgenommen werden.

4.1.3 Eröffnungsverfahren

Wie bereits erwähnt können die deterministischen Heuristiken aus Kapitel 3 als Eröffnungsverfahren verwendet werden. Dies hat den Vorteil, dass bereits sehr gute Startlösungen vorliegen, die weiter optimiert werden können. Für viele der Verfahren ist es jedoch von Vorteil, wenn zusätzlich zufällige Lösungen erzeugt werden. Dies erhöht die Diversität innerhalb der Lösungsanwärter und erlaubt eine peniblere und breiter gefächerte Suche im Lösungsraum auf Kosten der Konvergenzgeschwindigkeit. Die Erzeugung dieser zufälligen Lösungen kann ebenfalls von deterministischen Informationen Gebrauch machen, um die Qualität der Startlösungen zu verbessern, welches der langsamen Konvergenz stark entgegenwirkt. Für die hier verwendeten bionischen Optimierungsverfahren wurden die folgenden Eröffnungsverfahren basierend auf der Idee der Tiefen- bzw. Breitensuche entwickelt. Dabei werden sukzessive Cluster gebildet, indem Knoten zufällig hinzugefügt werden, welche sich in der Nachbarschaft der bisherigen Knoten des Clusters befinden. Ein Cluster ist „vollständig“, wenn kein Knoten ohne Verletzung der Zulässigkeit hinzugenommen werden kann.

Tiefensuche bzw. Nachfolgereröffnung:

Es wird zunächst ein zufälliger Knoten a des Graphen als Startknoten bestimmt. Von dort ausgehend wird ein zulässiger Nachfolgertausch (siehe Abschnitt 4.1.2) gesucht und durchgeführt. Dieser neu bestimmte Nachfolger wird nun wiederum als Startknoten a gewählt und die Suche nach einem Nachfolgertausch wird so lange fortgesetzt, bis kein zulässiger Knoten mehr gefunden werden kann. Auf diese Weise wird ein erstes Cluster mit sukzessivem Nachfolgertausch generiert. Nun wird erneut ein zufälliger Knoten a gewählt, jedoch muss hier darauf geachtet werden, dass a bisher noch nicht als Startknoten gewählt wurde bzw. Teil eines bereits bestehenden Clusters ist. Es wird erneut ein Cluster auf die soeben beschriebene Weise gebildet. Dies wird so lange wiederholt, bis kein unbetrachteter Knoten mehr existiert, der als Startknoten a gewählt werden könnte. Diese Eröffnungsstrategie ist exemplarisch an zwei Schritten in Abbildung 17 dargestellt.

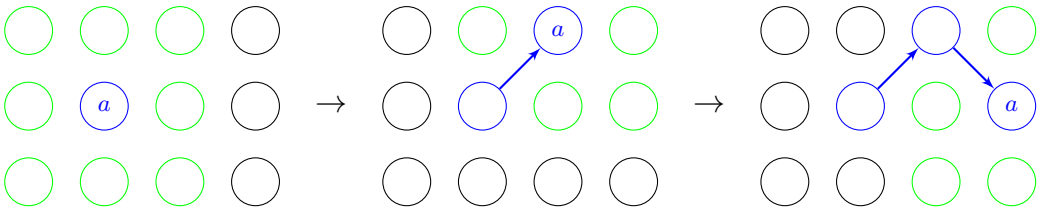


Abbildung 17: *Tiefensuche, exemplarisch dargestellt an zwei Schritten. Die grün markierten Felder kommen jeweils als neue Nachfolger in Frage.*

Breitensuche bzw. Vorgängereröffnung:

Ähnlich der Tiefensuche werden bei der Breitensuche Cluster durch suk-

zessive Nachbarschaftssuche erstellt. Wie bereits durch den Namen impliziert, werden hierbei alle benachbarten Knoten des bisherigen Clusters betrachtet, welche durch Hinzufügen die Zulässigkeit der Harkvorschrift nicht verletzen. In diesem Fall wird nicht vom Knoten a weggeharkt, sondern es wird zum Knoten a geharkt. Dies sorgt dafür, dass sich der Startknoten a

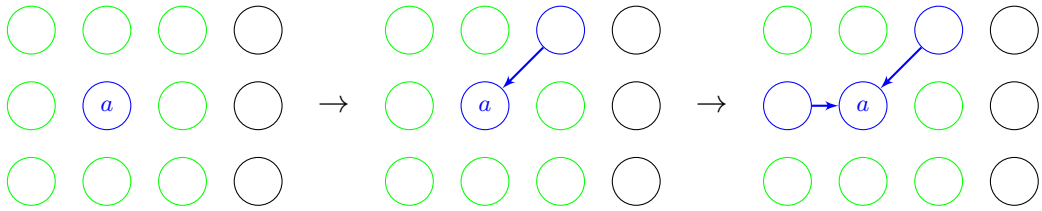


Abbildung 18: *Breitensuche, exemplarisch dargestellt an zwei Schritten. Die grün markierten Felder kommen jeweils als neue Vorgänger in Frage.*

erst einmal nicht ändert, sondern zunächst so viele Vorgänger wie möglich für diesen Knoten gesucht werden. Ist die Zunahme eines weiteren Knotens nicht möglich, da bereits alle Nachbarn als Vorgänger eines Knotens deklariert wurden oder durch Hinzunahme die Zulässigkeit verletzt würde, so werden sukzessive alle Vorgänger von a als neue Startknoten a gewählt. Sollte aufgrund der Kapazitätsbeschränkung kein weiterer Knoten mehr aufgenommen werden können, so gilt dieses Cluster als abgeschlossen und es wird ein neuer, noch nicht betrachteter Knoten, als Startknoten a gewählt. Auch hier werden auf diese Weise Cluster generiert, bis kein unbetrachteter Knoten mehr zur Auswahl steht. Dies ist in Abbildung 18, exemplarisch an zwei Schritten, dargestellt.

Die hier vorgestellten Eröffnungsverfahren lassen sich als spezielle Ausprägungen der in Abschnitt 3.1 beschriebenen sukzessiven Clusterbildung auffassen, indem die Wahl sowohl der Repräsentaten als auch der Aufnahmekandidaten nach dem Zufälligkeitsprinzip vorgenommen wird; bei der Nachfolgereröffnung (Tiefensuche) erfolgt zudem eine einfache und schnelle „Straight-forward-Suche“, also ohne ein „Backtracking“, d.h. eine (Rückwärts-)Suche nach weiteren möglichen Aufnahmekandidaten.

Im Anschluss an die Breiten- bzw. Tiefensuche kann wiederum eine explizite Hubbestimmung für die neuen Cluster nach Abschnitt 3.3 vorgenommen werden.

4.2 Ein genetischer Lösungsansatz

Genetische Algorithmen sind von der Evolution inspirierte Metaheuristiken zur Optimierung verschiedenster Problemstellungen. Die ersten genetischen Algorithmen wurden in den 1950er Jahren implementiert [2, 20]. Wie die meisten gän-

gigen bionischen Optimierungsverfahren folgt auch der genetische Algorithmus dem in Abschnitt 4.1 dargestellten Grundprinzip. Die Besonderheit der genetischen Algorithmen findet sich im dritten Schritt, der Erschaffung einer neuen Population. Nachdem im zweiten Schritt die besten Lösungsanwärter, im Kontext der genetischen Algorithmen *Chromosome* genannt, bestimmt wurden (*Selektion*), können neue Chromosome durch *Kreuzung* zwei vorhergehender Chromosome erzeugt werden. Darauffolgend werden diese *Nachfahren* durch zufällige *Mutation* geringfügig verändert. Diese Nachfahren bilden nun nach diesem Prozess die neue Population. Häufig ist es jedoch üblich, das beste Chromosom der vorhergehenden Population ebenfalls zu übernehmen. Dies sorgt dafür, dass das beste Chromosom der Nachfolgepopulation schlechtestenfalls genauso gut wie das der vorhergehenden ist.[60]

4.2.1 Selektion

Zur Bewertung eines Chromosomes betrachten wir eine Fitnessfunktion $F(s)$, wobei s das Chromosom bzw. der Lösungsanwärter ist. Da diese Funktion proportional zur Güte verläuft, resultiert das in einen antiproportionalen Verlauf zur Kostenfunktion $K(s)$ (2.28). Da die Fitnessfunktion zur Berechnung von Überlebenswahrscheinlichkeiten genutzt wird, ist es von großem Vorteil, eine strikt positive Fitnessfunktion zu definieren. Da in den betrachteten Szenarien $K(s) > 0$ gilt, eignet sich das Inverse der Kostenfunktion als Fitnessfunktion:

$$F(s) = \frac{1}{K(s)}. \tag{4.3}$$

Das k-te Chromosom der Population P^i zum Iterationsschritt i wird mit s_k^i bezeichnet. Im Folgenden wird $F(s_k^i)$ durch F_k^i abgekürzt. Es sei m die Populationsgröße und $\overline{F^i}$ die Summe über alle Fitnesswerte einer Population P^i:

$$\overline{F^i} = \sum_{k=1}^{m} F_k^i. \tag{4.4}$$

Für die Erzeugung einer neuen Population, bestehend aus zufällig bestimmten Chromosomen der vorhergehenden Population, sei eine normierte Fitnessfunktion

$$N(s_k^i) = N_k^i = \frac{F_k^i}{\overline{F^i}} \tag{4.5}$$

gegeben. Die akkumulierte normierte Fitnessfunktion $\hat{N}(s)$ sei definiert als

$$\hat{N}(s_k^i) = \hat{N}_k^i = \sum_{j=1}^{k} N_j^i. \tag{4.6}$$

[60] Dies beschreibt einen *elitären genetischen Algorithmus*.

Zur einfacheren Bestimmung der nächsten Population gelte zusätzlich $\hat{N}_0^i = 0$.

Es existieren verschiedene Ansätze zur Selektion, die im Folgenden beschrieben werden.

Rouletteradselektion mit einem Zeiger (R1)

Sei X eine stetig gleichverteilte Zufallsvariable mit Realisation in $[0, 1)$, dann ergibt sich daraus für jedes Chromosom $\tilde{s}_k^i$ der folgenden Zwischenpopulation und einem zugehörigen Zufallswert x_k^i verteilt nach X

$$\tilde{s}_k^i := s_n^i, \text{ falls } \hat{N}_{n-1}^i \leq x_k^i < \hat{N}_n^i. \tag{4.7}$$

Die Wahrscheinlichkeit, dass ein beliebiges Chromosom s (ein oder mehrmals) in die folgende Zwischenpopulation übernommen wird, ergibt sich somit als

$$W_{R1}(s) = 1 - (1 - N(s))^m. \tag{4.8}$$

Die **R1**-Selektion hat hohen Selektionsdruck[61], da die Fitnessfunktion einen sehr hohen Einfluss auf die Überlebenswahrscheinlichkeit jedes Chromosomes hat, es ist jedoch möglich, dass auch schlechte Chromosome mehrfach in der folgenden Zwischenpopulation vorkommen. Des Weiteren ist es nicht sichergestellt, dass das beste Chromosom überlebt (nicht elitär), daher muss es künstlich hinzugefügt werden.

Rouletteradselektion mit m Zeigern (RM)

Sei X eine stetig gleichverteilte Zufallsvariable mit Realisation in $[0, \frac{1}{m})$, dann ergibt sich daraus für jedes Chromosom $\tilde{s}_k^i$ der Zwischenpopulation und einem allgemeinen Zufallswert x^i verteilt nach X

$$\tilde{s}_k^i := s_n^i, \text{ falls } \hat{N}_{n-1}^i \leq x^i + \frac{k-1}{m} < \hat{N}_n^i. \tag{4.9}$$

Die Wahrscheinlichkeit, dass ein beliebiges Chromosom s (ein oder mehrmals) in die folgende Population übernommen wird, ergibt sich in diesem Fall als

$$W_{RM}(s) = \min\{N(s) \cdot m, 1\}. \tag{4.10}$$

Die Besonderheit der **RM**-Selektion besteht darin, dass Chromosome mit einem normierten Fitnessfunktionswert $N(s) \geq \frac{1}{m}$ in jedem Falle in die neue Zwischenpopulation aufgenommen werden, daher ist diese Selektionsmethode elitär. Des Weiteren ist es nur diesen auch möglich, mehrfach in

[61] Hoher Selektionsdruck bedeutet, dass weniger gut angepasste Individuen eine sehr geringe Überlebenschance haben [10, 17].

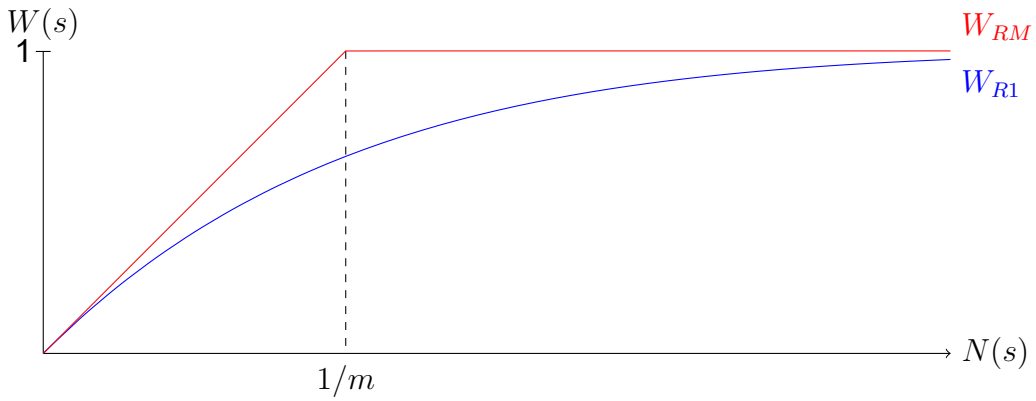


Abbildung 19: *Vergleich der Auswahlwahrscheinlichkeiten der Selektionsmethoden **R1** und **RM**.*

der neuen Zwischenpopulation vorzukommen. Die maximale und minimale Häufigkeit eines Chromosomes lässt sich ebenfalls bestimmen:

$$H_{RM}^{max}(s) = \lceil N(s) \cdot m \rceil \tag{4.11}$$

$$H_{RM}^{min}(s) = \lfloor N(s) \cdot m \rfloor \tag{4.12}$$

Bei der **RM**-Selektionsmethode ist der deterministische Anteil dominanter, da die Fitnessgüte die mögliche Häufigkeit der überlebenden Individuen eines Chromosoms stark einschränkt. Dahingegen ist der stochastische Anteil bei **R1** stärker und jedes Chromosom kann sich so häufig duplizieren, wie es der Zufall erlaubt. In Abbildung 19 lässt sich erkennen, dass der Selektionsdruck niedriger für **RM** ausfällt, da Chromosome jeder Fitnessgüte eine höhere Überlebenswahrscheinlichkeit haben. Dies wirkt zunächst kontraintuitiv, bedeutet jedoch nur, dass für **R1** eine höhere Wahrscheinlichkeit für generelles Aufkommen von Duplikaten besteht, sodass andere Chromosome gar nicht überleben.

4.2.2 Kreuzung

Beim Prozess der Kreuzung werden mehrfach zwei Chromosome s_k und s_l aus der selektierten Zwischenpopulation bestimmt, um die Informationen dieser beiden Chromosome zu vereinen. Bei einer Kreuzung entstehen ein oder mehrere neue Chromosome, die jedoch keine neuen Informationen beinhalten, sondern nur bereits Bekanntes neu kombinieren. Dieses Vorgehen ist sehr problemspezifisch, da darauf geachtet werden muss, dass potentielle Nebenbedingungen an die Lösungsanwärter nicht verletzt werden.

Für die Optimierung des Laubharkproblems wurde das Verfahren der **Hubkreuzung** entwickelt und implementiert. Die jeweilige Bestimmung von zwei Chro-

mosomen s_k und s_l geschieht zufällig, aus der Kreuzung entstehen zwei neue Chromosome. Dabei soll jedes Chromosom genau einmal ausgewählt werden. Bei der *Rouletteradselektion mit einem Zeiger* können der Einfachheit halber „benachbarte“ Chromosome zur Kreuzung genutzt werden, d.h. wähle $l = k + 1$ für ungerade k. Wird eine andere Selektionsmethode gewählt, die keine zufällige Anordnung erzeugt, werden zufällig Chromosomenpaare gewählt.

Der Informationsaustausch zwischen den beiden zur Kreuzung ausgewählten Chromosome geschieht durch den Austausch der Positionen zweier zufällig bestimmter Hubs h_k in s_k und h_l in s_l. Da die Digraphen per Definition zyklenfrei

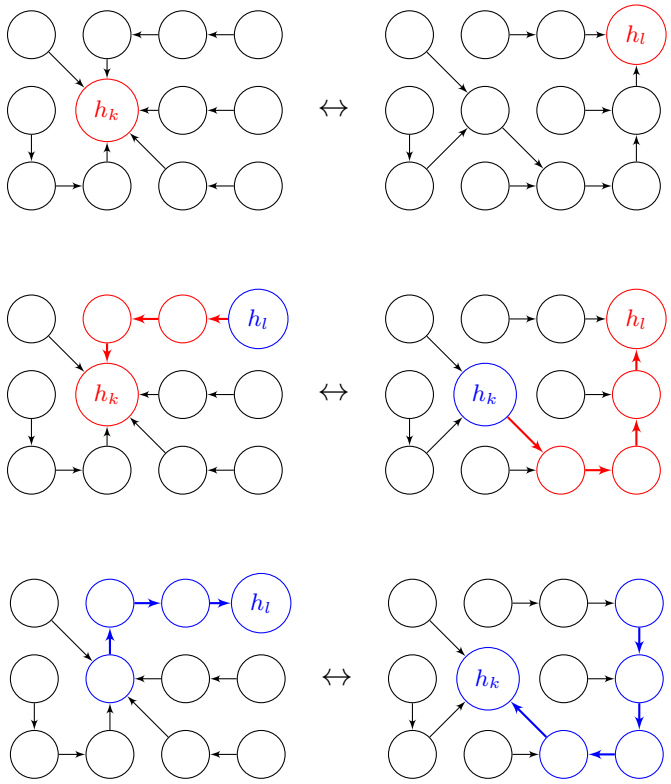


Abbildung 20: *Hub-Kreuzung zweier Chromosome durch Pfadinvertierung, vereinfach dargestellt an zwei Mono-Cluster-Chromosomen.*

sein müssen, existiert in jedem Chromosom mindestens ein Hub. Der Hub h_k wird nun künstlich in s_l injiziert und vice versa, dies ist exemplarisch in Abbildung 20 dargestellt. Um dies zu realisieren, wird der Pfad des Hub-Anwärter-Knotens zum aktuellen Hub des Clusters gesucht und invertiert. Dies wandelt automatisch der vorherige Hub in einen regulären Knoten um. Anschließend wird der Anwärter als sein eigener Nachfolger definiert, was ihn per Definition zu einem Hub macht. Die Hubkreuzung bringt mehrere Vorteile mit sich, allem voran ist es nicht nötig, die Nebenbedingungen zu überprüfen. Die Kapazitätsbeschränkung $\overline{M}$ kann nicht verletzt werden, da das Cluster nicht verändert wird.[62] Des Weiteren kann eine

[62] Unter der Voraussetzung, dass die Generalisierung der Kapazitätsbeschränkung $\overline{M}(a) = \overline{M}$ für alle $a \in V$ gilt.

Pfadinvertierung, die zwangsläufig in einem Hub endet, aus trivialen Gründen keinen Zyklus erzeugen.

4.2.3 Mutation

Die Mutation wird für jedes Chromosom der Zwischenpopulation nach der Kreuzung unter einer bestimmten Mutationswahrscheinlichkeit durchgeführt und ist eine direkte Anwendung der Suche in der Nachbarschaft, wie definiert in Abschnitt 4.1.2. In allen Tests wurde die Nachbarschaftssuche mithilfe des Nachfolgertausches verwendet. Für jedes in Frage kommende Chromosom wird ein zufälliger Knoten bestimmt, dessen Nachfolger geändert werden soll. Alle möglichen Anwärter werden (in zufälliger Reihenfolge) kontrolliert, ob sie sich als Nachfolger eignen (unter Einhaltung der Zyklenfreiheit und Kapazitätsgrenze). Ist ein passender gefunden, so wird dieser Nachfolgertausch durchgeführt. Da jeder Knoten auch als sein eigener Nachfolger und somit zu einem Hub deklariert werden kann, ist dies immer eine zulässige Mutation. Es wird nicht überprüft, ob der neue Nachfolger sich tatsächlich vom alten Nachfolger unterscheidet.

Genetischer Algorithmus

Eingabe: Fitnessfunktion fit, zulässiger Suchraum S
Parameter: Populationsgröße $popsize$
Initialisierung: Startpopulation P^0, Iterationenzähler $i = 0$
Bewerte[a] P^0

while nicht Abbruchbedingung **loop**
    $i := i + 1$
    Setze $\tilde{P}^i := \emptyset$
    for j in $\{1, ..., popsize\}$ **loop**
        Auswahl des i-ten Chromosoms[b] für $\tilde{P}^i$ aus P^{i-1}
    end for
    Erzeuge $popsize$ Nachkommen in P^i mit Anwendung genetischer
    Operatoren[c] auf $\tilde{P}^i$
    Bewerte P^i
end while

Ausgabe: Generation P^i

[a] Berechnung der Fitnesswerte
[b] Selektion basierend auf der Fitness
[c] Jedes Chromosom aus $\tilde{P}^i$ hat die Möglichkeit zur „Kreuzung“ und „Mutation“

4.3 Lösung mithilfe eines Bienenalgorithmus

Schwarmbasierte Optimierungsverfahren haben sich in den letzten Jahren in vielen Anwendungsbereichen etabliert. Bekannte Vertreter bilden hierbei die Ameisenalgorithmen, das Partikelschwarmverfahren, der Glühwürmchenalgorithmus, aber auch der Bienenalgorithmus. Diese bionischen Verfahren sind sehr vielseitig, flexibel und oft auch einfach zu implementieren [22].

Im Rahmen des Laubharkproblems wurde der Bienenalgorithmus implementiert, welcher auf dem sogenannten Schwänzeltanz der Bienen basiert. Der Schwänzeltanz ermöglicht im Rahmen der Nahrungssuche, die besten Futterstellen ausfindig zu machen. Generell wird die Nahrungssuche durch sogenannte Scout-Bienen organisiert. Die Scout-Bienen suchen in der Umgebung des Bienenstocks nach reichhaltigen Nahrungsquellen. Nach erfolgreicher Suche kommunizieren die Scout-Bienen mit den sogenannten Sammelbienen über den Schwänzeltanz und übermitteln insbesondere die Entfernung und die Richtung der Futterquelle. Die rekrutierten Sammelbienen fliegen nun zur Futterquelle, um die Nahrung zu ernten. Bei besonders reichhaltigen Futterquellen können auch die Sammelbienen den Schwänzeltanz ausführen und damit weitere Bienen für die gefundene Futterquelle rekrutieren. Grundsätzlich werden die rekrutierten Bienen nicht exakt an dieselbe Stelle fliegen, welche ursprünglich von der Scout-Biene gefunden wurde. Dazu ist die Ortsangabe durch den Schwänzeltanz zu ungenau. Bei nahrungsreichen Futterquellen wird die Sammelbiene aber auf jeden Fall auf Nahrung stoßen und auch weitere Bienen für diese Futterquelle gewinnen können.

Dieses Verhalten bei der Nahrungssuche wurde algorithmisch folgendermaßen aufgearbeitet [15]. Zuerst schwärmen ns Scout-Bienen zufällig aus, um Futterquellen ausfindig zu machen. Die gefundenen Positionen werden bewertet und es werden die nb besten sowie ne „allerbesten“ Futterquellen für die weitere Suche bestimmt. Daraufhin werden jeweils nre Bienen rekrutiert, um die Nachbarschaft der allerbesten Futterstelle abzusuchen. Des Weiteren werden jeweils nrb Bienen für die Nachbarschaft der noch übrig gebliebenen besten $nb - ne$ Futterstellen rekrutiert. Falls die zu einer Futterstelle rekrutierten Bienen in der Nachbarschaft einer Futterstelle eine nahrungsreichere Quelle gefunden haben, wird diese unter einer vordefinierten Wahrscheinlichkeit bei der nächsten Iteration anstatt der ursprünglichen Futterstelle berücksichtigt. Zu guter Letzt wird eine globale Suche mit dem Aussenden von $ns - nb$ Scout-Bienen angestoßen, welche wie zu Beginn zufällig im Raum ausschwärmen (vgl. Pseudocode des Algorithmus auf Seite 55).

Der Bienenalgorithmus kann auf beliebige Optimierungsprobleme angewendet werden, falls für diese folgende Fragen und algorithmische Schritte geklärt sind:

1. Wie sieht die Codierung des Problems bzw. des Lösungsraums aus?
2. Wie bestimmt man einen beliebigen Punkt im Lösungsraum?
3. Wie wird die Güte eines Punktes im Lösungsraum bewertet?
4. Wie ist die Nachbarschaft eines Punktes im Lösungsraum definiert?

Diese Fragen wurden für das Laubharkproblem bereits in den vorangegangenen Kapiteln und Abschnitten beantwortet. Die Codierung eines Punktes im Lösungsraum wird durch die Darstellung des Harkprozesses durch die Nachfolgerfunktion (vgl. 2.2.2) realisiert. Dabei ist darauf zu achten, dass diese nicht nur zyklenfrei, sondern auch im Sinne der Definition zulässig ist, d.h. dass durch den induzierten Harkprozess die maximal zulässige Laubmenge auf einem Feld nicht überschritten wird. Einen beliebigen Punkt im Lösungsraum erhält man entweder durch die Anwendung eines der intuitiven Heuristiken aus Kapitel 3 oder durch eines der Eröffnungsverfahren aus Abschnitt 4.1.3 (Tiefensuche bzw. Nachfolgereröffnung, Breitensuche bzw. Vorgängereröffnung). Die Güte eines Punktes im Lösungsraums ergibt sich durch die Berechnung des Harkaufwands, der unproduktiven Wegezeiten sowie des Transportaufwands (vgl. Kapitel 2). Für das Auffinden eines weiteren Punktes in der Nachbarschaft eines bisherigen Punktes im Lösungraum dient die Nachbarschaftsdefinition in Abschnitt 4.1.2 (Nachfolgertausch, Clustertrennung bzw. Clustervereinigung).

Durch die Parameter ns, nb, ne, nre, nrb lässt sich auf das Konvergenzverhalten des Bienenalgorithmus Einfluss nehmen und entweder die globale oder die lokale Suche stärker gewichten. Die Wahl der Parameter hat auch einen direkten Einfluss auf die benötigte Rechenzeit. Als Abbruchbedingung dient einerseits die maximale Anzahl an Iterationsschritten als auch die Tatsache, dass die bisher beste gefundene Lösung sich über mehrere Iterationen nicht mehr ändert (Stagnation).

Bienen-Algorithmus - Grundsätzlicher Ablauf

Eingabe: Zielfunktion f [a]

Parameter: Scout-Bienen-Anzahl ns
Futterstellen-Anzahl nb, ne [b]
Bienen-Anzahl nrb, nre [c]

Initialisierung: Verteilung der Scout-Bienen

Bestimme anhand der ns Fitnesswerte die nb und ne „besten“ Stellen

while $t < t_{max}$ bzw. Stagnation **loop**
  for alle zu ne gehörigen Bienen **loop**
    Bestimme nre Stellen in der jeweiligen Nachbarschaft
    Wähle lokal beste Stelle als Ort der neuen Scout-Biene oder akzeptiere unter einer bestimmten Wahrscheinlichkeit eine schlechtere Position
  end for
  for alle zu $nb - ne$ gehörigen Bienen **loop**
    Bestimme nrb Stellen in der jeweiligen Nachbarschaft
    Wähle lokal beste Stelle als Ort der neuen Scout-Biene oder akzeptiere unter einer bestimmten Wahrscheinlichkeit eine schlechtere Position
  end for
  Verteile $ns - nb$ Scout-Bienen zufällig im Raum
  Bestimme anhand der ns Fitnesswerte die nb und ne „besten“ Stellen
  $t := t + 1$
end while

Ausgabe: Beste bisher gefundene Lösung

[a] Berechnung der Fitness-Werte
[b] Stellen mit besten Fitness-Werten
[c] Die jeweils zu rekrutierenden Bienen

5 Das Programm lhp

Das Programm `lhp` zur Bearbeitung und zum Lösen von Laubharkproblemen ist vollständig in MATLAB verfasst. Es beinhaltet Funktionen zum Einlesen, zur manuellen Erstellung oder randomisierten Erzeugung von Gärten, zur Bedienung und Verwaltung der Algorithmen sowie Durchführung von vergleichenden Tests.

In Abschnitt 5.1 findet eine textbasierte Einführung in den MATLAB-Code zur Arbeit an Laubharkproblemen in Skripten und im Command Window statt. Sie richtet sich vornehmlich an eine Leserschaft, welche mit dem Programmieren in MATLAB vertraut ist und sich für die programmtechnische Struktur und den Aufbau des Programms interessiert, beispielsweise zwecks Weiterentwicklung des Programms (ggf. zum Einbau weiterer Algorithmen).

Zum komfortablen Arbeiten mit dem MATLAB-Programm bieten sich allerdings auch grafische Benutzeroberflächen (Graphic User Interfaces $\equiv$ GUI) an, die in Form eines Manuals in Abschnitt 5.2 ausführlich vorgestellt werden. Demnach richtet sich dieser Abschnitt vornehmlich an eine Leserschaft, die das MATLAB-Programm `lhp` via Benutzeroberflächen zur Lösung von „eigenen“ Laubharkproblemen benutzen möchte, wozu Abschnitt 5.1 lediglich als „Nachschlagewerk“ dienen mag.

5.1 Aufbau des Programms

In diesem Abschnitt findet eine ausführliche Einführung in den MATLAB-Code des Programms `lhp` statt. Der Schwerpunkt wird dabei auf das systematische Testen und Vergleichen der Algorithmen gelegt anstatt auf die Erklärung der implementierten Algorithmen selbst. Auf eine ausführliche Darstellung aller Bedienszenarien sowie eine Beleuchtung aller Parameter für alle entwickelten Funktionen wird aus Gründen des Textumfangs bewusst verzichtet. Die Leserschaft wird stattdessen auf die eingebaute Dokumentation der Quellcodes verwiesen, die in MATLAB mit den Befehlen `help` und `doc` abrufbar ist:

```
% help ist besonders nuetzlich fuer Funktionen
help lhp.Garden.random
```

```
% doc ist besonders nuetzlich fuer Klassen
doc lhp.ProblemData
% oder zum Nachschlagen von Methoden
doc lhp.ProblemData.plot
```

Das gesamte Programm ist als MATLAB-Modul mit dem Namen `lhp` entwickelt worden. Alle im Folgenden vorgestellten Inhalte gliedern sich in die hierarchische Struktur dieses Moduls ein. Auf diese Weise kann man auch ohne vorherige Kenntnis des Programms anhand der Modulpfade erahnen, welche Funktionen die enthaltenen Programme ausüben. Zum Beispiel umfasst der Code im Modul `lhp.algo` Algorithmen zur Lösung des Laubharkproblems. Dort, wo fest definierte Datenstrukturen eine Menge an Operationen auf sich vereinen, wurde ein objektorientiertes Entwicklungsschema verwendet (Beispiel: Die Klasse `lhp.Garden` beschäftigt sich exklusiv mit Gärten). Auch diese Klassen sind in die Modulstruktur integriert.

Der Code, auf den sich dieser Text bezieht, ist mit MATLAB® Release 2020b (Update 5) entwickelt worden. Eine Kompatibilität mit anderen Programmversionen wird nicht garantiert.

Darüber hinaus werden folgende *Toolboxen* zur Arbeit mit dem Projekt benötigt:

- Parallel Computing Toolbox
- varycolor (zur Darstellung einiger Plots im `testScript.mlx`)
- Statistics and Machine Learning Toolbox (für `testScript.mlx`)

Dieser Abschnitt beschäftigt sich ausschließlich mit der code-basierten Bedienung des Programms. Die folgenden Erklärungen und Beispiele sollen die Leserschaft dazu befähigen, im MATLAB Command-Window oder in eigenen MATLAB-Skripten mit dem Laubharkprojekt arbeiten zu können.

Allgemeine Erläuterungen zur Bedienung des Programms werden mit Codebeispielen vervollständigt. In diesem Sinne empfiehlt es sich, bei der Lektüre dieses Abschnitts den Quellcode in MATLAB zu öffnen, um die Beispiele dort selbst nachzuvollziehen. Darüber hinaus kann so bei Unklarheiten im Umgang mit dem Code die Dokumentation zu Rate gezogen werden. Alternativ ist im Quellcode ein MATLAB Live Script hinterlegt, das die gleichen Inhalte wie dieser Text in interaktiver Form bereithält.

5.1.1 Erzeugung eines Gartens

Ausgehend von der Theorie zum Laubharkproblem ist ein Garten ein ungerichteter, kanten- und knotenbewerteter Graph. In Anlehnung an den Vorgang des

realen Laubharkens ist in diesem Text von *Feldern* die Rede. Graphentheoretisch betrachtet sind immer Knoten gemeint. Ein Garten ist nun eine Matrix, die beschreibt, wie viel Laub auf je einem rechteckigen Abschnitt (*Feld bzw. Knoten*) des Gartens liegt. Die Werte der Felder des Gartens haben die folgende Bedeutung:

- Feld > 0: Auf diesem Feld liegt die angegebene Menge Laub.
- Feld $= 0$: Auf diesem Feld liegt kein Laub.
- Feld $= -1$: Dieses Feld ist durch einen Gartenschuppen blockiert.
- Feld $= -2$: Dieses Feld ist durch einen Baum blockiert.
- Feld $= -10$: Dieses Feld ist durch den Komposthaufen blockiert.

Das Problem ist so ausgelegt, dass über blockierte Felder (Laubmenge < 0) nicht gegangen und auch nicht geharkt werden kann. Die Kennzeichnung dient vor allem der Unterscheidbarkeit zwischen blockierten Feldern. So ist zum Beispiel immer klar, wo in einem Garten das Kompostfeld ist (Laubmenge $= -10$). Die Bewertungsfunktionen gehen davon aus, dass genau ein Kompostfeld in einem Garten existiert. Die Algorithmen interpretieren die negativen Laubmengen nicht weiter. Insofern können beliebige weitere Markierungen für blockierte Felder definiert werden.

Um die Darstellung der Gartenmatrizen zu vereinheitlichen, ist die Klasse `Garden` im `lhp`-Modul entwickelt worden. Diese Klasse bündelt die Gartenmatrix mit einer Menge fest definierter Operationen, die auf einem Garten durchgeführt werden können, wie zum Beispiel das Erzeugen oder grafische Darstellen von Gärten.

Um einen Garten zu erzeugen, wird die Funktion `lhp.Garden.random()` aufgerufen. Als notwendige Parameter erwartet die Funktion die Anzahl an Zeilen und Spalten des gewünschten Gartens. Der folgende Aufruf erzeugt einen Garten mit 15 Zeilen und 15 Spalten:

```
Garten = lhp.Garden.random(15, 15)
```

Mit Hilfe weiterer *Key-Value*-Parameter kann die Funktion modifiziert werden. Der Parameter `'Export'` speichert den erzeugten Garten in einer Datei ab, der Parameter `'Plot'` erzeugt zu dem Garten einen heatmap plot (siehe Abbildung 21):

```
lhp.Garden.random(15, 15, "Plot", true)
```

Das Kompostfeld hat in dieser Ansicht den Wert -10 und ist schwarz eingefärbt. In der Gartenmatrix lauten die Werte wie oben beschrieben. Bei der Erzeugung des Gartens wird darauf geachtet, dass keine eingeschlossenen Laubfelder existieren können, d.h. der entsprechende Graph muss zusammenhängend sein. Sollten eingeschlossene Felder vorhanden sein, wird der Garten so lange neu

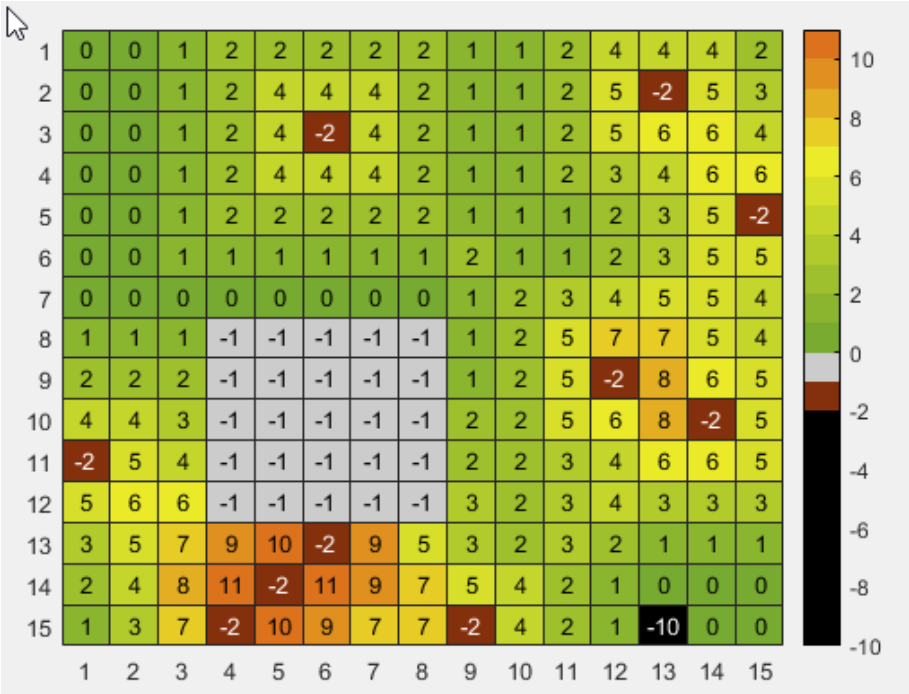


Abbildung 21: *Plot eines Beispielgartens.*

erstellt, bis alle laubbesetzten Felder erreichbar sind.

Um mit dem Garten arbeiten zu können, werden einige Metadaten benötigt:

- Ein Startknoten, an dem mit der Arbeit begonnen wird (`Start`).
- Eine maximale Laubmenge, die auf einem Feld aufgehäuft werden kann (`Max_Val`).
- Die maximale Transportkapazität der Schubkarre (`Max_Trans`).
- Die Kostenfaktoren für das Harken, Transportieren und unproduktive Wege (`Rake_Param`, `Trans_Param`, `Unprod_Param`).
- Eine Beschreibung des Gartens (`GMatrix`).
- Eine Beschreibung der Nachbarschaften der Felder (`Adjancency`).
- Die Distanzen der Felder zueinander, unter Berücksichtigung blockierter Felder (`DMatrix`).

Die `GMatrix` ist eine 4-spaltige Matrix, deren erste und zweite Spalte jeweils die Zeile und Spalte eines Gartenfeldes angeben. Die dritte Spalte definiert den Index des Feldes bei zeilenweiser Zählung und die vierte Spalte beinhaltet die Laubmenge des Feldes. Die Adjazenzmatrix (`Adjacency`) beschreibt, welche Felder, unter Berücksichtigung blockierter Felder, direkt miteinander benachbart sind. Die Distanzmatrix (`DMatrix`) beinhaltet die Entfernungen aller Felder zueinander, berechnet nach Dijkstras Algorithmus.

5.1.2 Beschreibung eines Gartens mit zugehörigen Metadaten

Ein Objekt des Typs `ProblemData` stellt den Garten samt zugehöriger Metadaten zur Verfügung. Auf diese Weise beschreibt ein Objekt der `ProblemData`-Klasse immer vollständig eine konkrete Problemstellung und ist die zentrale Datenstruktur, mit der alle Algorithmen arbeiten.

Um eine einfache Erzeugung von Problemen zu ermöglichen, kann der zu verwendende Garten von der `ProblemData`-Klasse „zufällig“ erzeugt werden. Dazu gibt man, wie zuvor, die gewünschte Anzahl an Zeilen und Spalten an:

```
lhp.ProblemData(15, 15)
```

Um einen zuvor erstellten Garten in ein Objekt vom Typ `ProblemData` einzubetten, wird folgende Syntax verwendet:

```
pdata = lhp.ProblemData(15, 15, "Garten", Garten)
```

Die ersten beiden Parameter sind auch hier zwingend erforderlich. Bei dieser Art des Aufrufs wird die Anzahl an Zeilen und Spalten aber objektintern mit den entsprechenden Werten für den übergebenen Garten überschrieben.

Darüber hinaus bietet die `ProblemData`-Klasse weitere einstellbare Parameter. Dabei gilt es zu beachten, dass eine Manipulation der enthaltenen Gartenmatrix (`pdata.Original`) in einem erzeugten Objekt nicht erlaubt ist. Eine derartige Manipulation würde automatisch zu einer Änderung der Adjazenz-, Distanz- und GMatrix führen. In diesem Fall ist es besser, mit dem manipulierten Garten ein neues Objekt vom Typ `ProblemData` zu erzeugen.

Zwei Parameter in der Ausgabe sollen noch besonders hervorgehoben werden. Der Parameter `DiagonalWeight` bestimmt die Kosten (als Wegeinheit) für das diagonale Gehen. Ein Wert von ∞ verbietet das diagonale Gehen. Ein Wert von $\sqrt{2}$ entspricht der euklidischen Distanz. Der Parameter `flattened` ist nicht parametrierbar, sondern von der Struktur fest vorgegeben.

Da die Adjazenz-, Distanz- und GMatrix aus der Gartenmatrix berechnet werden können, müssen sie nicht zwingend abgespeichert werden. Insbesondere für große Gärten von ca. 70 x 70 Feldern ist der Speicherbedarf dieser Strukturen um einen Faktor 500 größer als der Speicherbedarf der anderen in `ProblemData` enthaltenen Parameter. Daher können Objekte vom Typ `ProblemData` „komprimiert“ werden, wobei die Adjazenz-, Distanz- und GMatrix gelöscht werden. Diese Komprimierung findet mit der Methode `flatten` statt:

```
pdata = pdata.flatten()
```

In diesem Zustand können die Algorithmen das Problem nicht lösen, da essenzielle Metadaten fehlen. Daher ist es nötig, ein komprimiertes Objekt vom Typ `ProblemData` vor der weiteren Verwendung zu „dekomprimieren“:

```
pdata = pdata.unflatten()
```

Dabei werden die fehlenden Metadaten von neuem berechnet und eingesetzt. Des Weiteren kann die enthaltene Gartenmatrix direkt grafisch dargestellt werden:

```
pdata.plot()
```

oder man lässt sich eine textuelle Beschreibung des Problems ausgeben, zum Beispiel im Command Window:

```
pdata.print()
```

oder in einer Textdatei:

```
fd = fopen("beispielaufruf_pdata_print.txt", "w");
pdata.print(fd);
fclose(fd);
```

Hierbei ist jedoch zu beachten, dass algorithmen-spezifische Parameter nicht mit ausgegeben werden.

Die `ProblemData`-Klasse akzeptiert weitere Konstruktor-Key-Value-Parameter, um sehr spezifische Probleme erzeugen zu können. Mit dieser Problembeschreibung als Grundlage kann mit der Betrachtung der Algorithmen zur Lösung der Probleme fortgefahren werden.

5.1.3 Deterministisch-heuristische Lösungsverfahren

Die deterministischen Lösungsverfahren, enthalten im Modul `lhp.algo.deterministic`, erzeugen auf deterministische Art und Weise Lösungen für das Laubharkproblem. Dabei lassen sich zwei übergeordnete Algorithmentypen unterscheiden:

- Die simultanen Clusterverfahren im Modul `lhp.algo.deterministic.simultaneous_cluster`,
- die sukzessiven Clusterverfahren im Modul `lhp.algo.deterministic.successive_cluster`.

Die zusätzlich im Modul `lhp.algo.deterministic` enthaltene `zickzack`-Funktion stammt aus dem Projekt vorangegangenen Überlegungen zur Lösung des Laubharkproblems. Jedes der Verfahren ist samt seiner Funktionsweise im Quellcode dokumentiert, daher wird an dieser Stelle auf eine Erörterung der Algorithmen

verzichtet (vgl. Kapitel 3).

Alle Verfahren haben ein fest definiertes Interface: Sie erhalten ein Objekt vom Typ `ProblemData` und erzeugen daraus eine Nachfolgerfunktion `s`, die die ermittelte Harkvorschrift zur Lösung des Problems beschreibt. Da die Grundalgorithmen (`simultaneous_cluster, successive_cluster`) noch weitere Parameter zur Festlegung interner Entscheidungskriterien erhalten, sind für jeden der Grundalgorithmen spezifische Wrapper-Funktionen implementiert worden, die die Anforderungen an das Interface erfüllen.

Die aktuelle Programmversion umfasst 83 verschiedene deterministische Heuristiken. Zur Ermittlung, welche der Heuristiken besonders gute Ergebnisse liefern, ist ein eigenes Testszenario entwickelt worden, dass im `testScript` zu finden ist. Eine Heuristik lässt sich zwar direkt aufrufen:

```
s = lhp.algo.deterministic.zickzack(pdata)
```

Allerdings ist diese Art des Aufrufs nicht vorgesehen. Um eine Kompatibilität zu den stochastischen Heuristiken herzustellen (siehe Abschnitt 5.1.4), die unter anderem weitere Rückgabeparameter als die deterministischen Heuristiken liefern, gibt es die Wrapper-Klasse `DeterministicWrapper`, die von der Klasse `BaseWrapper` erbt. Daher sollten diese Heuristiken in die Wrapper-Klasse „gewickelt“ werden, die zusätzliche Komfortfunktionen liefert, wie gleich ersichtlich wird:

```
import("lhp.algo.*");
lhp_zz = DeterministicWrapper(@deterministic.zickzack, "Zickzack")
```

Um eine Berechnung durchzuführen, wird ein `ProblemData`-Objekt übergeben:

```
[index, result] = lhp_zz.add_new_data(pdata)
```

Die Variable `result` enthält eine Datenstruktur mit den Ergebnissen der Berechnung. Alle auf diese Weise durchgeführten Berechnungen werden objekt-intern gespeichert. Der Rückgabewert `index` gibt einerseits an, wie viele Ergebnisse bereits gespeichert worden sind. Auf der anderen Seite kann mit diesem Index das gewünschte Ergebnis aus der internen Ergebnistabelle abgerufen werden:

```
table_row = lhp_zz.get(index)
```

Ein Aufruf der `get()`-Methode ohne Parameter liefert die gesamte Tabelle zurück. Dazu werden zunächst noch einige Ergebnisse eingefügt:

```
for idx = 1:5
    lhp_zz.add_new_data(lhp.ProblemData(15+idx, 15+idx));
end
lhp_zz.get()
```

Alternativ lassen sich auch gezielte Bereiche ausgeben:

```
lhp_zz.get([2, 4])
```

Um sich die Anzahl an Ergebnissen in der Tabelle nicht merken zu müssen, kann mit `Inf` als Ersatz für das `end` in Arrays das letzte Element der Tabelle indexiert werden:

```
lhp_zz.get(Inf)
```

Da oftmals nur ein Bruchteil der Daten einer so erhaltenen Tabellenzeile benötigt wird, kann man die zu extrahierenden Daten weiter spezifizieren:

```
lhp_zz.get(2, "Kosten")
```

Welche Daten genau auf diesem Wege extrahiert werden können, verrät die Dokumentation zu der Methode:

```
help lhp_zz.get
```

Um die Ergebnisse der deterministischen Heuristiken für Problemstellungen, in denen das Verhältnis der maximalen Laub-Transportmenge und der maximalen Laubmenge pro Feld ungünstig gewählt ist, zu verbessern, kann die Laubmengenobergrenze optimiert werden (vgl. Abschnitt 3.4). Bei dieser Laubmengenobergrenzen-Optimierung durchlaufen die deterministischen Heuristiken beim Lösen des Gartens eine Schleife, in der das Problem mit jeder ganzzahligen Laubmengenobergrenze im Intervall $[1, MaxLaub]$ gelöst und die beste Lösung gespeichert wird. Das so ermittelte Laubmengenobergrenzen-Optimum wird als zusätzlicher Wert in den „results“-Strukturen eingepflegt, damit die Ergebnisse nachgerechnet werden können.

Um die Laubmengenobergrenzen-Optimierung zu aktivieren, muss ein zusätzlicher Konstruktorparameter übergeben werden:

```
import("lhp.algo.*");
DeterministicWrapper(@deterministic.zickzack, ...
                     "Zickzack", "OptimizeMaxLaub", true)
```

Die errechneten Daten können mit der `plot`-Methode in Abhängigkeit der Gartengröße dargestellt werden:

```
figure();
ax = axes();
lhp_zz.plot(ax, "Kosten")
```

Analog zur `print`-Methode von `ProblemData` kann ein Objekt vom Typ `DeterministicWrapper` seine Lösung ebenfalls textuell darstellen. Dazu muss zwingend der gewünschte Durchlauf (`index`) angegeben werden:

```
lhp_zz.print(2)
```

Um nach einem Durchlauf alle vorhandenen Ergebnisse zu löschen und von neuem Berechnungen zu beginnen, wird die `clear`-Methode verwendet:

```
lhp_zz.clear()
lhp_zz.add_new_data(pdata)
```

Zuletzt kann zum Beispiel zur Erzeugung von Legendeneinträgen in Plots der eingangs übergebene Name des Algorithmus formatiert ausgegeben werden:

```
lhp_zz.get_name("Latex")
```

Darüber hinaus gibt es weitere Konstruktor- und Methodenparameter (siehe Dokumentation).

Da, wie eingangs erwähnt, 83 deterministische Heuristiken im Laubharkprogramm existieren, die alle ihrerseits unterschiedliche Funktionspointer verwenden und unterschiedliche Bezeichner haben, ist eine Komfortmethode integriert worden, die die schnelle und einfache Erzeugung von `DeterministicWrapper`-Objekten erlaubt (Anm.: Kapitel 6 beschränkt sich auf eine Auswahl (algorithmsneu.txt) davon!). Die Methode `gather()` arbeitet auf Basis der Textdatei `heuristics.txt` im Ordner `+lhp/+algo/+deterministic`, die für jede deterministische Heuristik eine textuelle Beschreibung des Funktionshandle und Algorithmennamen beinhaltet. Die Methode `gather()` liest, basierend auf individuellen Eingaben, die gewünschten Algorithmen aus der Datei aus und erzeugt dafür Objekte vom Typ `DeterministicWrapper`. Dazu muss man lediglich die ID des gewünschten Algorithmus kennen, die der Textdatei zu entnehmen ist:

```
all_deterministics = lhp.algo.DeterministicWrapper.gather(...
                    "Range", (5:end))
```

Um „im großen Stil“ mit den Algorithmen zu arbeiten, können der `gather()`-Methode auch Konstruktorparameter übergeben werden, die auf alle erzeugten Objekte angewendet werden:

```
lhp.algo.DeterministicWrapper.gather(...
                    "OptimizeMaxLaub", true, ...
                    "StoreResults", false, ...
                    "NameSuffix", "_TEST", ...
                    "Range", (5:10))
```

5.1.4 Bionische Lösungsverfahren

In der vorliegenden Progammversion existieren zwei bionische Lösungsverfahren:

- Der genetische Algorithmus im Modul `lhp.algo.stochastic.genetic` (vgl. Abschnitt 4.2),
- der Bienenschwarm-Algorithmus im Modul `lhp.algo.stochastic.bee` (vgl. Abschnitt 4.3).

Auf eine Erklärung der Funktionsweise der Algorithmen soll auch hier verzichtet werden. Stattdessen wird auf die externe Dokumentation hingewiesen.

Wie bereits erwähnt haben die bionischen Lösungsverfahren ein anderes Interface als die deterministischen Heuristiken. Dies ist insbesondere der Tatsache geschuldet, dass die hier implementierten bionischen Verfahren mit Populationen von Individuen arbeiten und in diesem Sinne mehr Rückgabeparameter als nur eine Nachfolgerfunktion s liefern können. Um diesem Umstand gerecht zu werden, stellt die Wrapper-Klasse `StochasticWrapper` eine Datenkapselung bereit, mit der die bionischen Algorithmen auf die gleiche Weise wie die deterministischen Heuristiken in `DeterministicWrapper` verwendet werden können. Hierdurch werden Implementierungsunterschiede verborgen und es kann mit einem einheitlichen Interface gearbeitet werden.

Daher sind viele der im Folgenden demonstrierten Methoden identisch zu denen der Klasse `DeterministicWrapper`. Ein Objekt eines bionischen Verfahrens erzeugt man wie folgt:

```
import("lhp.algo.StochasticWrapper");
import("lhp.algo.stochastic.*");
% Fuer den Bienenalgorithmus:
lht_bio = StochasticWrapper(@bee.bienenalgorithmus, ...
        "Bienen-Algorithmus", bee.BeeParameters())
% Fuer den genetischen Algorithmus:
% lht_bio = StochasticWrapper(@genetic.genetic, ...
          "Genetischer-Algorithmus", genetic.GeneticParameters())
```

Anders als bei den deterministischen Heuristiken ist hier der dritte Konstruktorparameter hervorzuheben. Die hier implementierten bionischen Algorithmen sind vielfältig parametrierbar. Um den individuellen Anforderungen jedes Algorithmus gerecht zu werden, gibt es Parameter-Klassen, die algorithmen-spezifische Optionen definieren:

```
params = lhp.algo.stochastic.bee.BeeParameters()
% Aendern eines Parameters
params.NumEliteSolutions = 6
```

Die Bedeutung diese Parameter ist der jeweiligen Dokumentation zu entnehmen. Da diese Aufrufe sehr lang sind, gibt es Komfortmethoden um eine „Standardva-

riante“ der Algorithmen zu instanziieren:

```
% Fuer den Bienenalgorithmus:
lht_bio = lhp.algo.stochastic.bee.wrapper()
% Fuer den genetischen Algorithmus:
%lht_bio = lhp.algo.stochastic.genetic.wrapper()
```

Anders als bei den deterministischen Heuristiken können dem Konstruktor diverse Parameter übergeben werden:

```
help lhp.algo.StochasticWrapper.StochasticWrapper
```

Um eine Berechnung durchzuführen, wird auch hier ein `ProblemData`-Objekt übergeben:

```
[index, result] = lht_bio.add_new_data(pdata)
```

Die Variable `result` enthält ein Struct mit den Ergebnissen der Berechnung. Alle auf diese Weise durchgeführten Berechnungen werden objekt-intern gespeichert. Der Rückgabewert `index` gibt einerseits an, wie viele Ergebnisse bereits gespeichert worden sind. Auf der anderen Seite kann mit diesem Index das gewünschte Ergebnis aus der internen Ergebnistabelle ermittelt werden:

```
table_row = lht_bio.get(index)
```

Ein Aufruf der `get()`-Methode ohne Parameter liefert die gesamte Tabelle zurück. Dazu werden erst noch einige Ergebnisse eingefügt:

```
% Dieser Schritt dauert eine ganze Weile.
% Mit dem Konstruktorparameter "NumWorkers" liesse er sich durch
% Parallelisierung erheblich beschleunigen.
for idx = 1:5
    lht_bio.add_new_data(lhp.ProblemData(15+idx, 15+idx));
end
lht_bio.get()
```

Alternativ lassen sich auch gezielte Bereiche ausgeben:

```
lht_bio.get([2, 4])
```

Um sich die Anzahl an Ergebnissen in der Tabelle nicht merken zu müssen, kann mit `Inf` als Ersatz für das `end` in Arrays das letzte Element der Tabelle indexiert werden:

```
lht_bio.get(Inf)
```

Da oftmals nur ein Bruchteil der Daten einer so erhaltenen Tabellenzeile benötigt wird, kann man die zu extrahierenden Daten weiter spezifizieren:

```
lht_bio.get(2, "Kosten")
```

Anders als bei den deterministischen Heuristiken ist hierbei zu beachten, dass für jeden Durchlauf der Monte-Carlo-Simulation ein einzelnes Ergebnis vorliegt. Wird die Methode `get()` ohne dritten Parameter aufgerufen, wird das „beste“ Ergebnis (geringste Gesamtkosten) aus allen Durchläufen ausgegeben. Um die Ergebnisse aller Durchläufe zu sehen:

```
lht_bio.get(2, "Kosten", "All")
```

Welche Daten genau auf diesem Wege extrahiert werden können, verrät die Dokumentation zu der Methode:

```
help lht_bio.get
```

Um die errechneten Daten schnell über der Gartengröße zu plotten, bietet sich die `plot`-Methode an:

```
fig = figure();
ax = axes();
lht_bio.plot(ax, "Kosten")
```

Man beachte, dass analog zur `get()`-Methode auch hier ein dritter Parameter angegeben werden kann, um z.B. die Mittelwerte aus allen Monte-Carlo-Durchläufen zu plotten:

```
lht_bio.plot(ax, "Kosten", "Average")
```

Analog zur `print`-Methode von `ProblemData`, kann ein Objekt vom Typ `StochasticWrapper` seine Lösung ebenfalls textuell darstellen. Dazu muss zwingend der gewünschte Durchlauf (index) angegeben werden:

```
lht_bio.print(2)
```

Die `print`-Methode verwendet in der Ausgabe für jeden Kostenfaktor den Wert der Lösung mit den geringsten Gesamtkosten. Eine Modifikation dieses Verhaltens muss daher direkt im Quelltext der Methode erfolgen.

Um nach einem Durchlauf alle vorhandenen Ergebnisse zu löschen und von neuem Berechnungen zu beginnen, wird die `clear`-Methode verwendet:

```
lht_bio.clear()
lht_bio.add_new_data(pdata)
```

Zuletzt kann zum Beispiel zur Erzeugung von Legendeneinträgen in Plots der eingangs übergebene Name des Algorithmus formatiert ausgegeben werden:

```
lht_bio.get_name("Latex")
```

5.1.5 Durchführung von Tests zum Vergleich von Algorithmen

Zur Durchführung von Tests zum Vergleich der Algorithmen dient der `TestManager`. Seine Aufgabe ist es, jeden durchzuführenden Test mit jedem zu testenden Algorithmus auszuführen und die Ergebnisse zugänglich zu machen. Dazu hat dieser eine interne Tabelle, in der die Zeilen die Testszenarien und die Spalten die zu testenden Algorithmen darstellen. Im einzelnen erfüllt er die folgenden Funktionen:

- Aufnahme beliebig vieler zu testender Algorithmen.
- Aufnahme beliebig vieler durchzuführender Testszenarien.
- Parallelisierte Durchführung aller vorbereiteten Tests.
- Zugriff auf den gesamten Ergebnisdatensatz.
- Zugriff auf bestimmte Teile des Ergebnisdatensatzes (Filterung der Daten).
- Speicherung und spätere Wiederherstellung mit den MATLAB-Befehlen `save` und `load`.

Durch das Code-Design ist es beispielsweise möglich, eine zuvor gespeicherte Instanz des `TestManagers` wiederherzustellen und neue Tests und Algorithmen einzufügen und ausführen zu lassen. Dabei ist zu beachten, dass nur diejenigen Tests ausgeführt werden, die noch nicht vorher ausgeführt worden sind. Außerdem ist es möglich, den `TestManager` im Betrieb zu unterbrechen und zu einem späteren Zeitpunkt an der abgebrochenen Stelle mit dem Testen fortfahren zu lassen. All diese Anwendungsszenarien werden in den folgenden Beispielen gezeigt.

Der Konstruktor des `TestManagers` ist immer leer:

```
tm = lhp.utils.TestManager()
```

Die Tests sind Objekte vom Typ `ProblemData`. Dabei stellt jedes Objekt einen eigenen durchzuführenden Test (Testszenario) dar. Nun werden einige zufällig erzeugte Tests eingefügt:

```
for idx = 1:5
    pdata = lhp.ProblemData(15, 15);
    info_str = sprintf("Testcase %d", idx);
    tm.addTestCase(pdata, info_str);
end
```

Das zweite Argument der Funktion ist ein optionaler `String`, mit dem das jeweilige Testszenario beschrieben werden kann. Das macht es in der Ergebnistabelle leichter, den Überblick über die Bedeutung der Tests zu behalten. Die Methode `get_results()` zeigt uns die gesamte Ergebnistabelle an:

```
tm.get_results()
```

Hier sind die eingefügten Testszenarien zu sehen. Es fehlen noch einige zu testende Algorithmen:

```
for alg = lhp.algo.DeterministicWrapper.gather("Range", 5:9)
    tm.addAlgorithm(alg);
end
tm.addAlgorithm(lhp.algo.stochastic.bee.wrapper())
```

Wird das Objekt in der Kommandozeile ohne Methode aufgerufen, erhält man weitere Informationen über das Objekt:

```
tm
```

Ebenso kann die Ergebnistabelle inspiziert werden:

```
tm.get_results()
```

Bisher liegen noch keine Ergebnisse vor. Sind die Vorbereitungen getroffen, kann das Testen beginnen. Wie bereits erwähnt werden die Tests parallelisiert ausgeführt. Spätestens hier ist die Parallel Computing Toolbox erforderlich:

```
tm.runAllTests()
```

Der Testmanager informiert während der Ausführung der Tests über den aktuellen Status. Der Ladebalken wird bewusst nur alle 2 Sekunden aktualisiert, um möglichst viele CPU-Ressourcen für das Ausführen der Tests übrig zu lassen.

Sobald die Ergebnisse vorliegen, kann mit der Auswertung der Daten begonnen werden. Das geht entweder per Hand direkt in der Ergebnistabelle oder mit den angebotenen Methoden zum Filtern der Daten. Dafür ist die Methode `extract()` zuständig. Die Methode ist komplex in der Bedienung, aber mächtig hinsichtlich des Nutzens zur Datenauswertung. Am besten ist es, sich hier mit der Dokumentation vertraut zu machen, bevor das nächste Beispiel eingeführt wird:

```
help tm.extract
```

Angenommen, man möchte die Gesamtkosten des Algorithmus „`Zickzack`“ für jeden Garten in jedem Test ermitteln.

- Der Algorithmus ist „`Zickzack`“.
- Die Variable ist „`Kosten`“, hier „`K`“.
- Die Daten werden nicht sortiert, weil es für ein und denselben Garten jeweils nur einen Test und damit nichts zum Sortieren gibt.
- Die Daten werden nach den verschiedenen Gärten gruppiert (Bezeichner für Gärten in `ProblemData` ist „`Garden`“).

Der Aufruf sieht dann wie folgt aus:

```
[data, grouped_by] = tm.extract("Zickzack", "K", ...
                                "GroupBy", "Garden")
```

Die erhaltenen Daten können direkt geplottet werden. Die Daten in `data(1, 1, 3)` gehören zu dem Test des dritten Gartens, also `grouped_by{3}`. Die Selektion der Daten lässt sich, zum Beispiel mit dem Parameter „`From`“, weiter verfeinern. So könnten Ergebnisse spezifisch für Gärten abgefragt werden, in denen der Index des Kompost (`ProblemData.Target`) kleiner als 100 ist:

```
selection = tm.where("Target", "<", 100)
[data, grouped_by] = tm.extract("Zickzack", "K", ...
                                "GroupBy", "Garden", ...
                                "From", selection)
```

Weiterführende Beispiele und sonstige Anwendungen des `TestManager` sind in der Datei `testScript` zu sehen, in welcher einige Testszenarien berechnet werden. Zum Abspeichern verwendet man den MATLAB-Befehl `save()`.

```
save mytm_save tm
clear tm
load mytm_save
tm.get_results()
```

Nach dem Speichern, Löschen und erneuten Laden des `Testmanagers` sind alle Informationen noch erhalten.

Von den Erläuterungen zu `ProblemData` sollte bekannt sein, dass die Metadaten (Distanz-, Adjazenz- und GMatrix) den großen Teil des Speichers benötigen. Zu diesem Zweck lassen sich alle Metadaten der `ProblemData`-Objekte in der Tabelle vor dem Speichern entfernen und später wiederherstellen:

```
tm.flatten();
save mytm_save_compressed tm
clear tm
load mytm_save_compressed
tm.unflatten();
tm.get_results()
```

Bei der Betrachtung der von MATLAB erzeugten Speicherdateien dürfte auffallen, dass die „komprimierte“ Variante weniger Speicher braucht. Gerade bei großen Gärten von 50 x 50 Feldern und mehr fällt dieser Unterschied sehr groß aus. Abschließend ist zu zeigen, dass ein späteres Hinzufügen neuer Tests und Algorithmen kein Problem für den `TestManager` darstellt:

```
tm.addAlgorithm(...
      lhp.algo.DeterministicWrapper.gather("Range", 12));
tm.addTestCase(lhp.ProblemData(15, 15));
tm.runAllTests();
```

```
tm.get_results()
```

Ausführliche Anwendungsbeispiele finden sich in der Datei testScript.mlx. Da die komplette Ausführung dieser Programm-Datei sehr lange dauern kann, sind die einzelnen Zwischenergebnisse dem Programm beigefügt.

5.2 Dokumentation zur grafischen Benutzeroberfläche

In diesem Abschnitt wird die Bedienoberflächen (GUI) vorgestellt, welche die Bearbeitung und das Lösen von Laubharkproblemen erleichtert.

5.2.1 GUI zum Erzeugen von Algorithmen

Das GUI zum Erzeugen (Instanziieren) von Algorithmen ist im Wesentlichen ein Wrapper um die in Kapitel 5 aufgeführten Programme und Verfahren zusammenzustellen und auszuführen. Es stellt über diese Klassen hinaus keine weiteren Features bereit. Die folgenden Abbildungen 22 und 23 zeigen das GUI jeweils zum Konfigurieren deterministischer und bionischer Algorithmen. Um die Codeduplizierung zu minimieren, ist die Konfiguration beider Algorithmentypen in ein einziges GUI kombiniert worden.

5.2.1.1 Aufruf des GUI

Das GUI befindet sich im Modul `lhp.gui.Algorithm`. Am einfachsten wird das GUI direkt über das MATLAB–Command–Window aufgerufen:

```
lhp.gui.Algorithm
```

Allerdings gilt es zu beachten, dass auf diese Weise kein ausführbarer Algorithmus erzeugt wird: Ein Druck auf den „OK“–Knopf übergibt den Algorithmus an die Methode `disp()`, welche ihn in textueller Form im Command–Window ausgibt.

Um dieses Verhalten zu modifizieren, gibt es den `'Callback'`–Parameter, der beim Aufruf des GUI spezifiziert werden kann. Bei dem Callback handelt es sich um ein `function_handle`, also um einen Funktionspointer, der beim Bestätigen eines Algorithmus mit dem erzeugten Objekt dieses Algorithmus aufgerufen wird. Der übergebene Callback akzeptiert daher exakt ein Funktionsargument. Um auf diese Weise den erzeugten Algorithmus einer Variablen im base–Workspace zuzuweisen, wird folgender Aufruf benötigt

```
lhp.gui.Algorithm("Callback", @(alg) assignin('base', ...
```

```
'Algorithmus', alg));
```

Dieser Aufruf weist den erzeugten Algorithmus einer Variablen `Algorithmus` im MATLAB base–Workspace zu.

Hinweis

An dieser Stelle mag man sich fragen, weshalb es sich hierbei nicht um das „Standardverhalten“ des GUI handelt, wenn man es ohne weitere Parameter aufruft. Hierfür gibt es zwei Gründe:

(1) Das versehentliche Überschreiben bereits existierender Variablen soll vermieden werden.
(2) Wenn man im Command–Window arbeitet (von wo man das GUI in der Regel aufruft), bietet sich die direkte Konfiguration der Algorithmen über `BaseWrapper`–Kindklassen an.

5.2.1.2 Deterministische Heuristiken

Das GUI ist in 5 Kernbereiche (*Panels*) aufgeteilt, wie in Abbildung 22 markiert.

Im Panel (1) wird über zwei Knöpfe der Algorithmentyp ausgewählt. In Abbildung 22 ist die Konfiguration der deterministischen Heuristiken (`DeterministicWrapper`) gezeigt. Die Konfiguration bionischer Algorithmen ist in Abbildung 23 gezeigt.

Das Panel (2) dient zur Auswahl der konkreten Heuristik, die erzeugt werden soll. Der Nutzer hat über die Tabs zwei verschiedene Interfaces zur Auswahl:

Tab *Auswahl*

Hier werden die deterministischen Heuristiken nach den übergeordneten Algorithmenklassen konfiguriert. Zum aktuellen Zeitpunkt existieren 3 Algorithmenklassen:

1. Zickzack (`zickzack`),
2. Sukzessive Clusterverfahren (`successive_cluster`),
3. Simultane Clusterverfahren (`simultaneous_cluster`)

Jede der Algorithmenklassen hat ihrerseits verschiedene Parameter zur weiteren Spezifizierung des Verhaltens. Diese werden dem Nutzer als Drop–Down–Menu präsentiert, nachdem eine Algorithmenklasse ausgewählt worden ist. In Abbildung 22 sieht man, dass die sukzessiven Clusterverfahren unter anderem einen Parameter „Puffer Typ“ erwarten, der mehrere Optionen bereit stellt.

Tab *Alle Algorithmen*

In diesem Tab (nicht in Abbildung 22 gezeigt) wird dem Nutzer ein Suchfeld zusammen mit einer Liste **aller** deterministischen Heuristiken präsentiert. Die Algorithmen sind nach einem festen Namensschema benannt, das in erster Linie der algorithmischen Unterscheidbarkeit und nicht der direkten Lesbarkeit dient.

Über das Suchfeld können die anzuzeigenden Algorithmen gefiltert werden, indem der Nutzer die Anfangsbuchstaben der Algorithmen eintippt, die er sehen möchte. Diese Art des Zugriffs ist insbesondere für erfahrene Benutzer gedacht, die den Bezeichner des gewünschten Algorithmen bereits kennen.

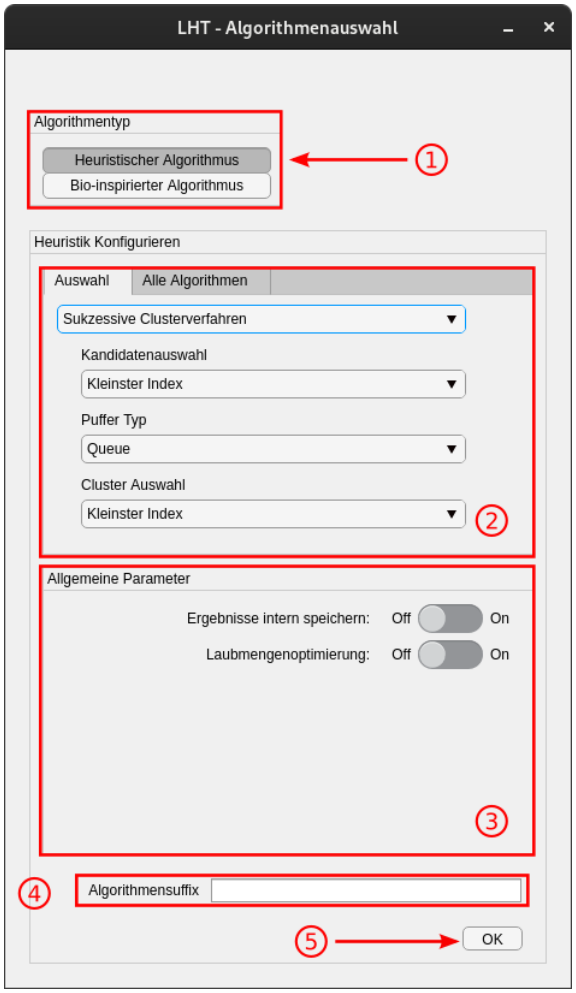


Abbildung 22: *GUI zum Konfigurieren deterministischer Algorithmen.*

Hinweis

Um die Suche durchzuführen, muss nach der Eingabe eines Textes die ‘Tab‘-Taste gedrückt oder mit der Maus irgendwo anders in das Fenster geklickt werden.

Im Panel (3) lassen sich Parameter einstellen, die für alle Algorithmen eines Typs gleichermaßen gültig sind. Aus Sicht des Programmcodes sind dies die Konstruk-

torparameter für DeterministicWrapper. Bei der Verwendung der Algorithmen im Zusammenhang mit dem TestManager empfiehlt es sich, die interne Speicherung der Ergebnisse zu deaktivieren.

Der Grundname aller erzeugten Algorithmen ist Programmintern vorgegeben. Um eine Unterscheidbarkeit zwischen verschieden konfigurierten Varianten des selben Algorithmus zu ermöglichen, kann mit dem Freitext-Feld im vierten Panel ein optionales Namenssuffix vergeben werden.

Zuletzt kann mit dem Knopf unter (5) die Konfiguration des Algorithmus bestätigt werden. Was genau mit dem erzeugten Algorithmus passiert, hängt davon ab, welche Parameter dem GUI beim Aufruf übergeben wurden. Wurde das GUI durch das Betätigen eines Knopfes in einem anderen GUI geöffnet, ist vom Nutzer nichts weiter zu tun, da die GUIs untereinander korrekt kommunizieren.

5.2.1.3 Bionische Algorithmen

Die Ansicht ist auch hier in 5 Panels (Abbildung 23) aufgeteilt, die die gleiche Aufgabe erfüllen wie bei der Konfiguration der deterministischen Heuristiken (Abbildung 22). Daher ist die folgende Beschreibung auf die wesentlichen Unterschiede beschränkt.

Im Panel (2) wird der konkrete bionische Algorithmus ausgewählt, der verwendet werden soll. Zum aktuellen Zeitpunkt stehen der genetische und der Bienenalgorithmus zur Auswahl. Die Parameter zu den jeweiligen Algorithmen werden über die Schaltfläche „Algorithmenspezifische Optionen“ konfiguriert.

Im Panel (3) lassen sich Parameter einstellen, die für alle bionischen Algorithmen gleichermaßen gültig sind. Aus Sicht des Programmcodes sind dies die Konstruktorparameter für StochasticWrapper.

Das Panel (4) sowie der Knopf (5) funktionieren wie oben beschrieben.

5.2.2 GUI zum Erzeugen von ProblemData–Objekten

Das GUI zum Erzeugen von ProblemData–Objekten ist ein Wrapper um die ProblemData–Klasse sowie um die Funktion Garden.random() zur Erzeugung zufälliger Gärten. Es ist in der Modulstruktur unter lhp.gui.ProblemData zu finden und besteht im Wesentlichen aus 3 Tabs, in denen ein Garten erstellt werden kann, sowie einem links angeordneten Panel, in dem sonstige Problemparameter

konfiguriert werden. Diese einzelnen Bestandteile werden im Folgenden erläutert.

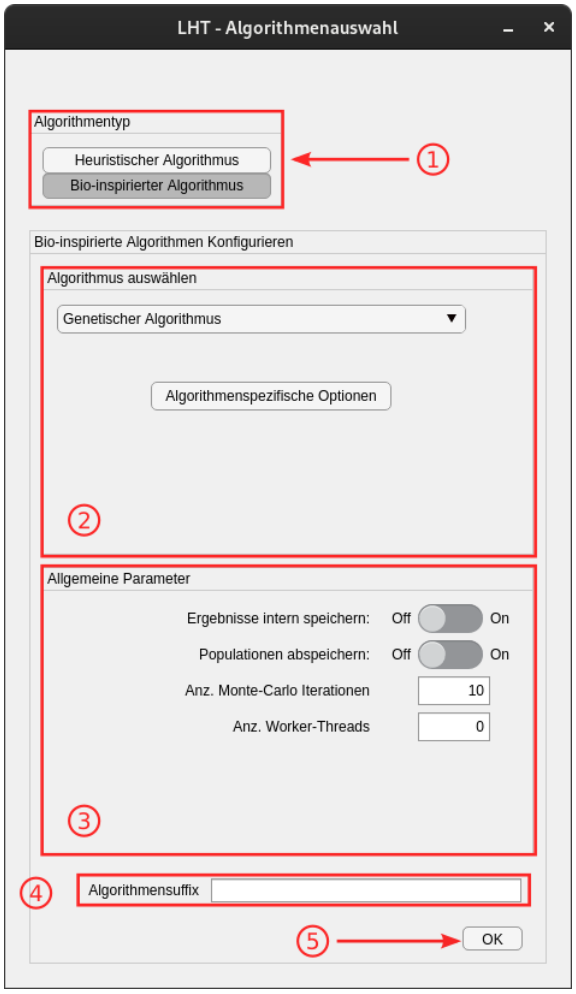


Abbildung 23: *GUI zum Konfigurieren bionischer Algorithmen*

5.2.2.1 Aufruf des GUI

Am einfachsten wird das GUI direkt über das MATLAB–Command–Window aufgerufen:

```
lhp.gui.ProblemData();
```

Im GUI gibt es diverse Möglichkeiten, bereits erzeugte „Probleme“ zu importieren. Es ist aber auch möglich, das GUI bereits beim Aufruf mit den Werten aus einem existierenden `ProblemData`–Objekt zu initialisieren:

```
% pdata muss ein Objekt vom Typ 'ProblemData' sein!
lhp.gui.ProblemData("ProblemData", pdata);
```

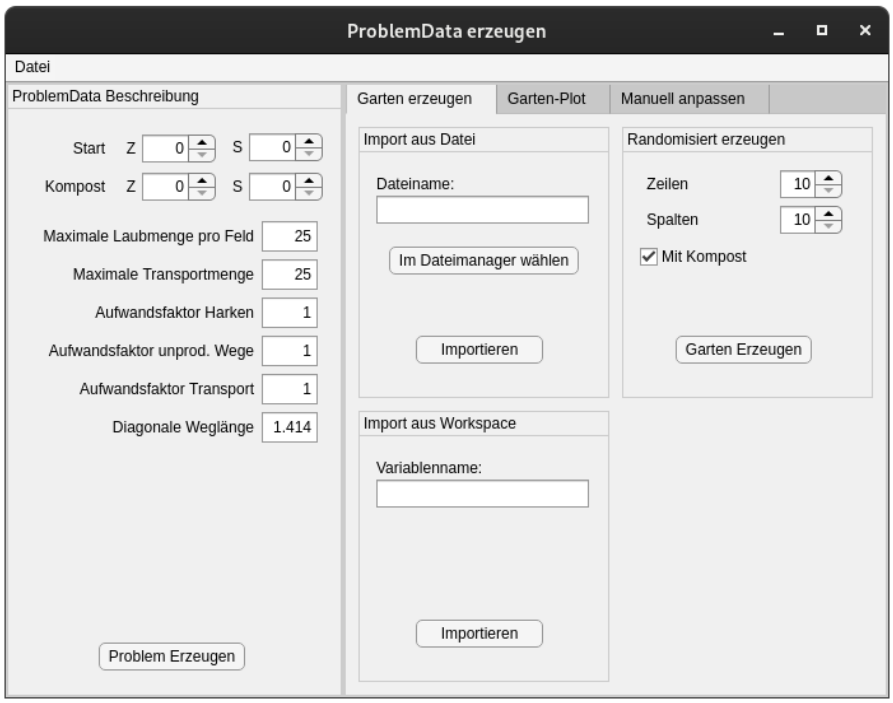


Abbildung 24: *Ansicht des GUI nach dem Öffnen.*

Es gilt zu beachten, dass bei diesem Aufruf kein gültiges Problem erzeugt wird: Ein Druck auf den „Erzeugen“–Knopf übergibt das erzeugte `ProblemData`–Objekt an die Methode `disp()`, die das Objekt in textueller Form im Command–Window ausgibt.

Um dieses Verhalten zu modifizieren, gibt es den `'Callback'`–Parameter, der beim Aufruf des GUI spezifiziert werden kann. Bei dem Callback handelt es sich um ein `function_handle`, also um einen Funktionspointer, der beim Bestätigen eines `ProblemData`–Objektes mit dem erzeugten Objekt aufgerufen wird. Der übergebene Callback akzeptiert daher exakt ein Funktionsargument.

Um auf diese Weise das erzeugte `ProblemData`–Objekt einer Variablen im base–Workspace zuzuweisen, wird folgender Aufruf benötigt:

```
lhp.gui.ProblemData("Callback", @(pdata) assignin('base', ...
                'Problem', pdata));
```

Dieser Aufruf weist das erzeugte Problem einer Variablen `Problem` im MATLAB–base–Workspace zu.

Hinweis

An dieser Stelle mag man sich fragen, weshalb es sich hierbei nicht um das „Standardverhalten“ des GUI handelt, wenn man es ohne weitere Parameter aufruft. Hierfür gibt es zwei Gründe:

1. Das versehentliche Überschreiben bereits existierender Variablen soll vermieden werden.
2. Wenn man im Command–Window arbeitet (von wo man das GUI in der Regel aufruft), bietet sich die direkte Konfiguration der Algorithmen über die `ProblemData`–Klasse an.

5.2.2.2 Hauptansicht des GUI

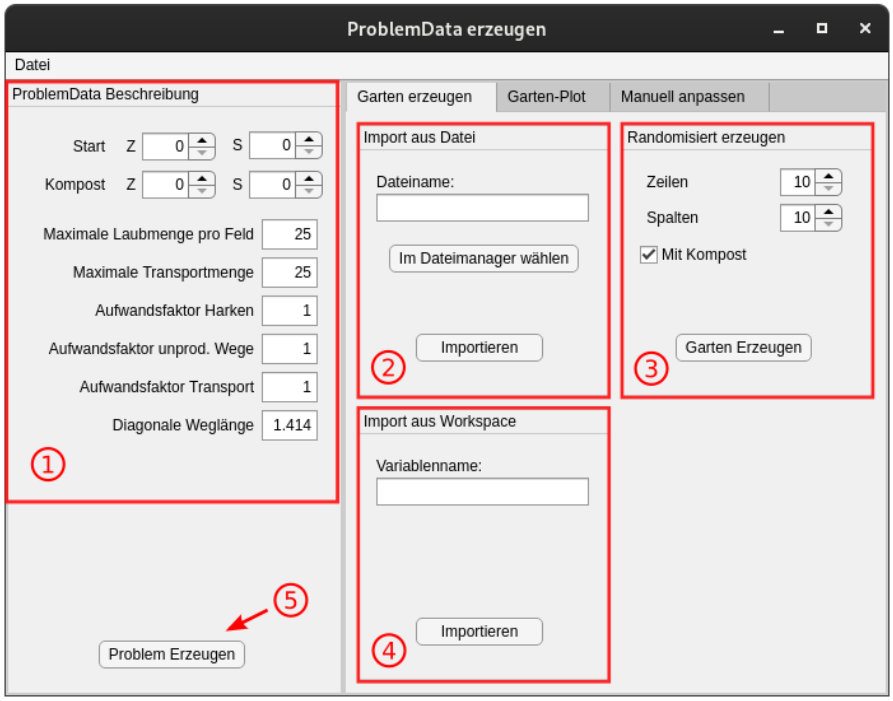


Abbildung 25: *Ansicht des GUI nach dem Öffnen.*

In Abbildung 25 ist die Hauptansicht des GUI unmittelbar nach dem Öffnen dargestellt. Es zeigt das Seitenpanel links, die Menüleiste oben (im Bild nicht markiert) sowie die Ansicht des ersten der 3 Tabs.
Markiert mit den Nummern (1) bis (4) sind verschiedene Panels, deren Funktion im Folgenden erläutert wird:
Das Panel (1) dient zur Konfiguration der Problemparameter. Das umfasst:

- Die Position des Start– und Kompostknotens
- Die Kosten für das Gehen, Harken und Abtransportieren von Laub
- Die maximale Laubmenge, die pro Feld aufgehäuft werden kann

- Die maximale Laubmenge, die in der Schubkarre auf einmal abtransportiert werden kann
- Der Kostenfaktor für diagonales Gehen und Harken

Hierbei sollte man berücksichtigen, dass zumindest die explizite Konfiguration eines Start– und Kompostknotens erst dann sinnvoll ist, wenn bereits ein Garten erzeugt wurde, da nicht alle Knoten dafür zulässig sind.

Hinweis

Wird die Voreinstellung von Start und Kompost bei `(0, 0)` gelassen, wird die Position beim Erzeugen (Schaltfläche (5)) des `ProblemData`–Objektes zufällig im Garten ausgewählt. Das ist dann nützlich, wenn die genaue Positionierung der Knoten keine große Rolle spielt.
Ausnahme beim Kompost: Ist im erzeugten Garten bereits ein Feld mit dem Wert -10, wird die Koordinate dieses Feldes als Kompoststelle übernommen!

Sollte hingegen bereits ein Garten mit Kompost vorhanden sein, und der Nutzer will den Kompost verschieben, muss er zuvor den bestehenden Kompost entfernen. Wie das geht, ist im Text unter Sektion 5.2.2.4 beschrieben.

Die Panels (2) bis (4) in Abbildung 25 verfolgen alle denselben Zweck: Die Bereitstellung eines Gartens. Der Nutzer hat somit drei verschiedene Möglichkeiten, einen Garten zu konfigurieren:

1. Der Garten kann aus einer `.mat`, `.xls(x)` oder `.csv` Datei eingelesen werden (Panel (2)).
2. Der Garten kann randomisiert erzeugt werden (Panel (3)).
3. Der Garten kann aus einer Variablen im base–Workspace eingelesen werden (Panel (4)).

In jedem Fall ist der erzeugte Garten hinterher im Tab *Garten–Plot* (Abbildung 26) zu sehen. Zuletzt erzeugt der Knopf mit der Nummer (5) in Abbildung 25 das konfigurierte `ProblemData`–Objekt.

5.2.2.3 Inspektion des erzeugten Gartens

Im zweiten Tab des GUI, in Abbildung 26 rot umrandet hervorgehoben, ist der erzeugte Garten grafisch dargestellt. Anhand der farblichen Unterscheidung der Felder des Gartens im Zusammenhang mit der Farbleiste kann der Nutzer direkt erkennen, welche Bedeutung die einzelnen Gartenfelder haben:

- Der Wert -10 markiert den Kompost (schwarz).
- Die Werte -2 markieren Bäume im Garten (braun).

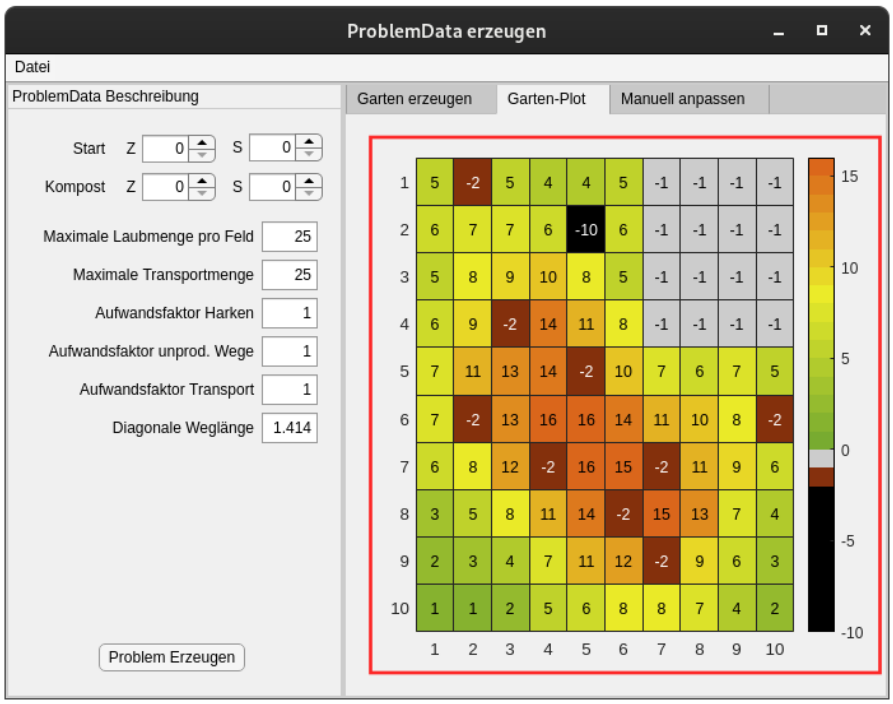


Abbildung 26: *Grafische Ansicht des erzeugten Gartens*

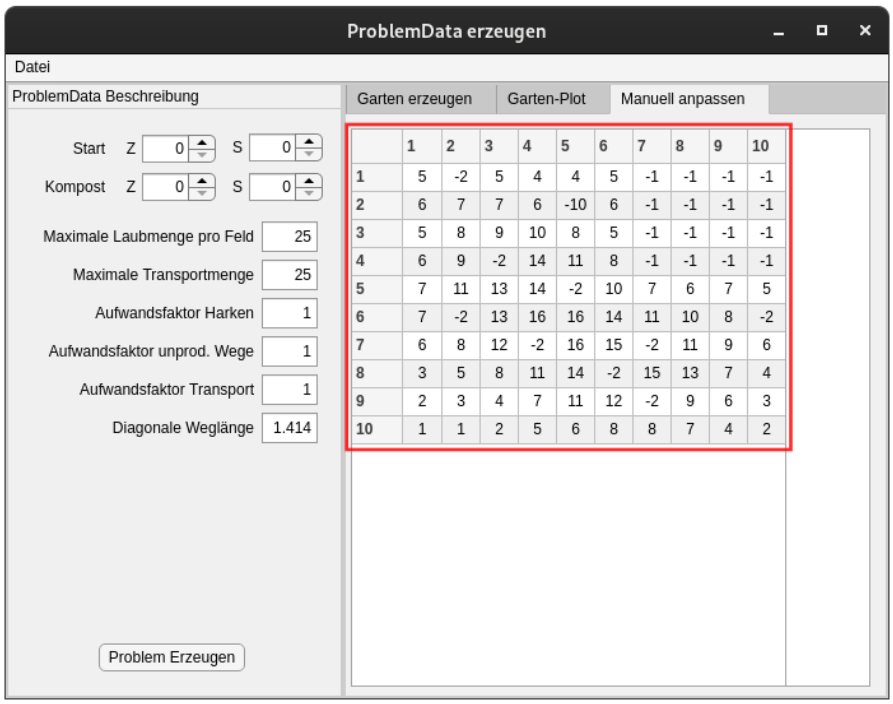
- Die Werte -1 markieren den Gartenschuppen (grau).
- Ein Wert 0 zeigt Felder ohne Laub (grün).
- Größere Werte werden von lindgrün über gelb zunehmend dunkler.

Der Gedanke hinter dieser Farbgebung ist, dass auf Feldern ohne Laub der (hoffentlich) grüne Rasen zu sehen ist. Je mehr Laub sich auf einem Feld sammelt, umso mehr nimmt es eine orange Färbung an.

Grundsätzlich können weitere Feldtypen mit negativen Zahlenwerten und einer zugehörigen Einfärbung definiert werden. In der aktuellen Programmversion hat diese Einteilung in unterschiedlich blockierte Felder (mit Ausnahme des Kompostfeldes) keinen Einfluss auf die Aufwandsberechnung.

5.2.2.4 Modifikation des erzeugten Gartens

Im dritten Tab des GUI (Abbildung 27) ist derselbe Garten wie im zweiten Tab (Abbildung 26) dargestellt, allerdings in Tabellenform. In dieser Ansicht können die Werte einzelner Felder des Gartens manuell angepasst werden. Das kann dann sinnvoll sein, wenn eine bestimmte Gartentopologie erzwungen werden soll. Wenn der Kompost an eine andere Stelle verschoben werden soll, ist eine händische Manipulation des Gartens sogar unerlässlich. Es ist in der aktuellen Programmversion nämlich nicht erlaubt, mehr als eine Kompoststelle im Garten zu haben!



	1	2	3	4	5	6	7	8	9	10
1	5	-2	5	4	4	5	-1	-1	-1	-1
2	6	7	7	6	-10	6	-1	-1	-1	-1
3	5	8	9	10	8	5	-1	-1	-1	-1
4	6	9	-2	14	11	8	-1	-1	-1	-1
5	7	11	13	14	-2	10	7	6	7	5
6	7	-2	13	16	16	14	11	10	8	-2
7	6	8	12	-2	16	15	-2	11	9	6
8	3	5	8	11	14	-2	15	13	7	4
9	2	3	4	7	11	12	-2	9	6	3
10	1	1	2	5	6	8	8	7	4	2

Abbildung 27: *Tabellarische Darstellung des Gartens*

Zum Ändern eines Wertes wird die entsprechende Zelle doppelt angeklickt. Daraufhin kann ein neuer Zahlenwert eingetragen werden. Bestätigt wird die Eingabe mit `Enter` oder durch Anklicken einer anderen Stelle des GUI. Die Ansicht im zweiten Tab (Abbildung 26) aktualisiert sich dann automatisch.

5.2.2.5 Import und Export

Zuletzt ist noch die Bedeutung der Menüleiste am oberen Fensterrand in Abbildung 25 zu erläutern. Durch einen Klick auf „Datei“ öffnet sich ein Menü, in dem zwischen „Import“ und „Export“ ausgewählt werden kann. Mit Hilfe dieses Menüs ist es möglich:

1. Erzeugte Probleme zu exportieren (Menüpunkt „Export“), und zwar
   a) als Variable in den Workspace („In den Workspace“), wobei der Name der zu erzeugenden Variablen dann vom Nutzer abgefragt wird;
   b) in eine `.mat`–Datei auf dem Rechner („In eine Datei“).
2. Zu modifizierende Probleme zu importieren (Menüpunkt „Import“), und zwar
   a) aus einer Variablen im Workspace („Aus dem Workspace“), wobei der Name der zu importierenden Variablen dann vom Nutzer abgefragt wird;
   b) aus einer `.mat`–Datei auf dem Rechner („Aus einer Datei“), in der ein Objekt vom `ProblemData` enthalten sein muss.

3. Zu modifizierende Gärten zu importieren (Menüpunkt „Import“), und zwar
   a) aus einer Variablen im Workspace (siehe oben, Variablentyp muss numerische Matrix sein);
   b) aus einer Datei (siehe oben, Datei darf in diesem Fall `.mat`, `.xls(x)` oder `.csv` sein).

Das Importieren von Garten- und `ProblemData`–Objekten ist hier doppelt belegt. Sofern der Nutzer eine `.mat`–Datei importiert, in der sowohl eine Gartenmatrix als auch ein `ProblemData`–Objekt enthalten ist, wird `ProblemData` bevorzugt.

5.2.3 GUI zum Erzeugen und Verwalten von `TestManager`–Objekten

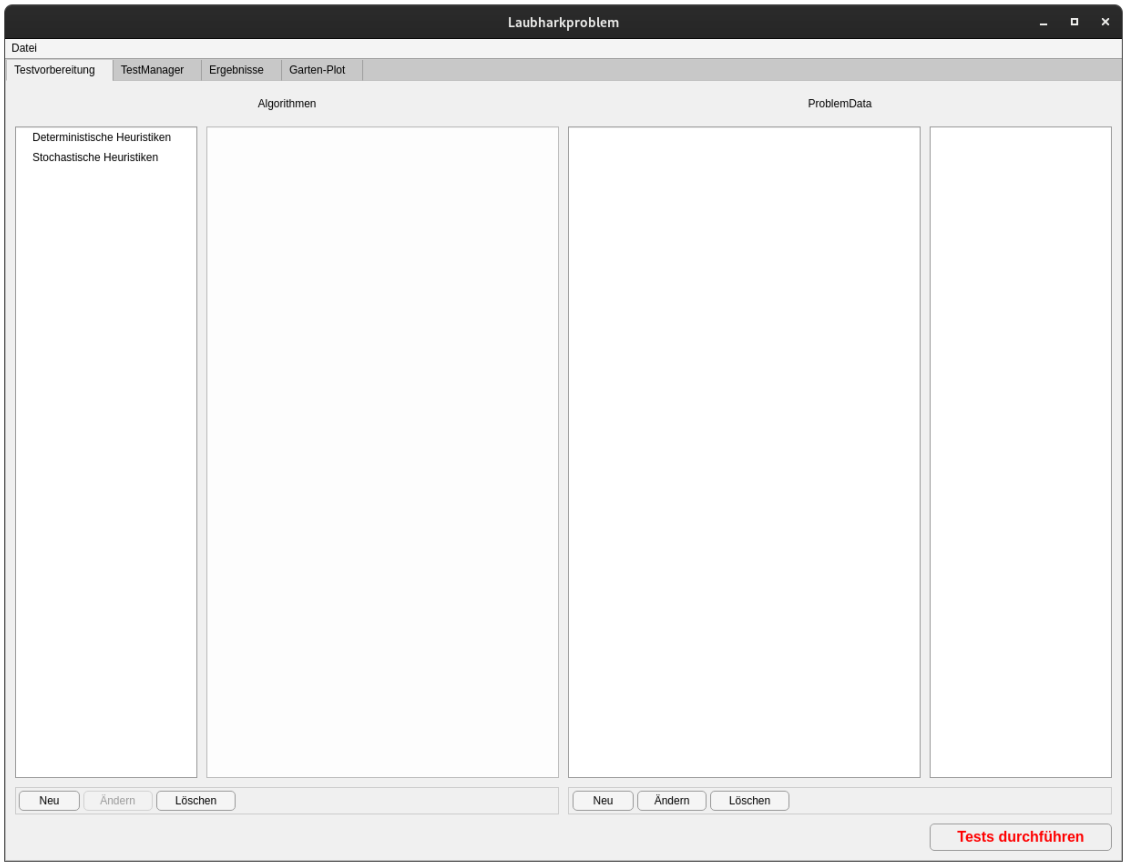


Abbildung 28: *Ansicht des GUI nach dem Öffnen.*

Das GUI zum Erzeugen und Verwalten von `TestManager`–Objekten (Abbildung 28) ist ein Wrapper um die `TestManager` Klasse. Es ist in der Modulstruktur unter `lhp.gui.TestManager` zu finden. Es besteht im Wesentlichen aus 4 Tabs, in denen ein `TestManager` erzeugt und verwaltet und die Visualisierung von Ergebnissen ermöglicht wird. Die einzelnen Bestandteile werden im Folgenden erläutert.

5.2.3.1 Aufruf des GUI

Am einfachsten wird das GUI direkt über das MATLAB–Command–Window aufgerufen:

```
lhp.gui.TestManager();
```

Im GUI gibt es diverse Möglichkeiten, bereits erzeugte `TestManager` zu importieren. Es ist aber auch möglich, das GUI bereits beim Aufruf mit den Werten aus einem existierenden `TestManager`–Objekt zu initialisieren:

```
% tm muss ein Objekt vom Typ 'TestManager' sein!
lhp.gui.TestManager(tm);
```

Um einen `TestManager` aus dem GUI abzuspeichern, wird die Menüleiste verwendet. Dies ist weiter unten erklärt.

5.2.3.2 Hauptansicht des GUI

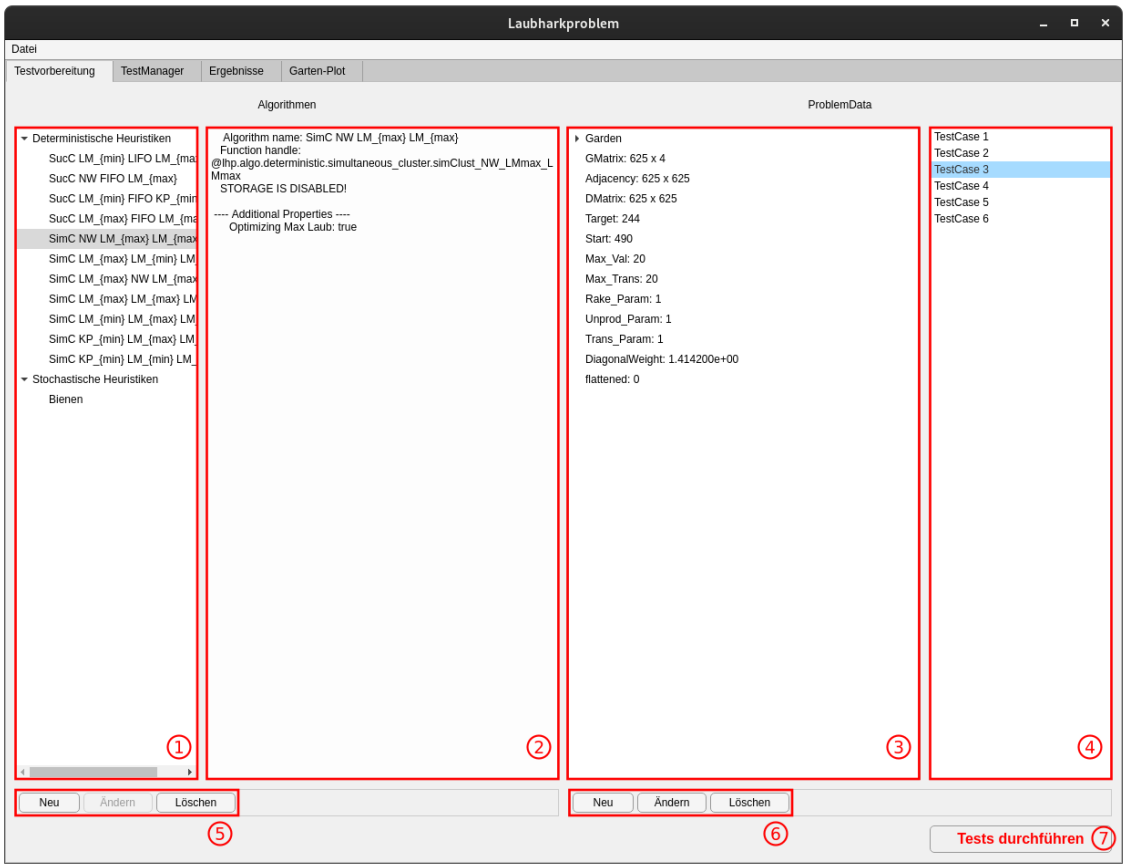


Abbildung 29: *Ansicht des GUI nach dem Import eines* `TestManager`.

In Abbildung 29 ist ein Screenshot des GUI gezeigt, nachdem ein bereits vorhandener `TestManager` importiert wurde. Es zeigt den ersten Tab („Testvorbereitung“), in den zu testende Algorithmen und Testszenarien verwaltet werden. Dieser Tab ist in mehrere Panels unterteilt. Panel (1), (2) und (5) dienen dem Verwalten

der zu testenden Algorithmen. Über die Bedienelemente im Panel (5) können neue Algorithmen hinzugefügt werden (Knopf „Neu“) sowie bereits vorhandene, im ersten Panel ausgewählte Algorithmen entfernt werden (Knopf „Löschen“).

Nach dem Hinzufügen eines neuen Algorithmus öffnet sich das GUI zum Erzeugen von Algorithmen. Wird dort ein Algorithmus über den Knopf „OK“ bestätigt, taucht er automatisch in der Liste in Panel (1) auf. Die Ansicht in diesem Panel ist so gestaltet, dass deterministische und bionische Heuristiken getrennt voneinander dargestellt werden.

Hinweis

Wird ein Algorithmus im GUI hinzugefügt, schließt sich das Fenster zum Konfigurieren der Algorithmen nicht von selbst. Das ist so beabsichtigt, weil es dem Nutzer ermöglicht, mehrere Algorithmen nacheinander zu konfigurieren, ohne den Dialog jedesmal von neuem zu öffnen.

Wählt der Nutzer in Panel (1) einen Algorithmus aus, erscheint in Panel (2) eine textuelle Beschreibung des Algorithmus. Diese Beschreibung dient dem schnellen Überblick über die bereits konfigurierten Algorithmen.
Panel (3), (4) und (6) werden zum Verwalten der durchzuführenden Testszenarien verwendet. Über die Bedienelemente in Panel (6) können

- neue Testszenarien hinzugefügt werden (Knopf „Neu“);
- bestehende Testszenarien geändert werden (Knopf „Ändern“);
- bestehende Testszenarien gelöscht werden (Knopf „Löschen“), nachdem sie in Panel (4) ausgewählt wurden.

Hinweis

Beim „Ändern“ von Testszenarien werden diese aus der Ergebnistabelle des `TestManager` gelöscht und als neuer Test (neue Zeile) in die Tabelle eingefügt. Hierbei wird die *TestCaseID* geändert!

Ein Betätigen des Knopfes „Neu“ öffnet das GUI zum Erzeugen von ProblemData–Objekten. Wird dort ein `ProblemData`–Objekt über den Knopf „Erzeugen“ bestätigt, taucht es in der Liste in Panel (4) auf.

Hinweis

Wird ein Testszenario im GUI hinzugefügt, schließt sich das Fenster zum Konfigurieren von `ProblemData` nicht von selbst. Das ist so beabsichtigt, weil es dem Nutzer ermöglicht, mehrere Testszenarien nacheinander zu konfigurieren, ohne den Dialog jedesmal von neuem zu öffnen.

Wählt der Nutzer in Panel (4) ein Testszenario aus, erscheint in Panel (3) eine textuelle Beschreibung des Szenarios. Diese Beschreibung soll dem Nutzer einen schnellen Überblick über die bereits konfigurierten Testszenarien bieten. Darüber hinaus wird die Gartenmatrix für den ausgewählten Test im vierten Tab („Garten–Plot“) grafisch dargestellt.

Zuletzt startet Knopf (7) die Berechnung aller noch nicht zuvor berechneten Testszenarien mit den entsprechenden Algorithmen. Einen Überblick darüber, welche Tests bereits durchgelaufen sind und welche nicht, bietet der zweite Tab des GUI, zu sehen in Abbildung 30 und 31.

5.2.3.3 Der „TestManager“–Tab

Der zweite Tab des GUI zeigt die aktuellen Ergebnisse des `TestManager`. Um verschiedenen Anforderungen bei der Auswertung der Ergebnisse gerecht zu werden, sind zwei verschiedene Darstellungsarten implementiert. Die Darstellungsart „Übersicht“ ist hauptsächlich zum schnellen Vergleich der Algorithmen über mehrere Tests hinweg geeignet. So lässt sich für mehrere Tests schnell feststellen, wie ein Algorithmus mit Blick auf die Minimierung von Zielvorgaben[63] im Vergleich zu anderen Algorithmen abgeschnitten hat. Auf der anderen Seite hilft die Darstellungsart „Pro Test“ beim Vergleich der Kostenbestandteile von Algorithmen für einen definierten Test.

Das Umschalten zwischen den Darstellungen findet jeweils über die Schaltfläche in Panel (2) in Abbildung 30 und 31 statt.

Darstellungsart „Übersicht“
Panel (1) zeigt die Ergebnisse in tabellarischer Form, wie sie die Methode `get_results()` des `TestManager` zurückgibt. Diese Ansicht ist interaktiv, eine Auswahl bestimmter Objekte zieht weitere Aktionen nach sich:

- **Auswahl einer Ergebnis–Zelle eines Algorithmus**
  Wird ein einzelnes Ergebnis (eine Zelle) eines Algorithmus markiert, wird im Tab „Ergebnisse“ die zugehörige Lösung angezeigt und der Tab „Garten–Plot“ zeigt eine grafische Darstellung des Gartens. Außerdem wird in Panel (5) eine textuelle Beschreibung des gelösten Problems dargestellt.

[63] Die bisher berücksichtigten Zielvorgaben sind: Harkkosten, Wegekosten, Transportkosten bzw. Gesamtkosten.

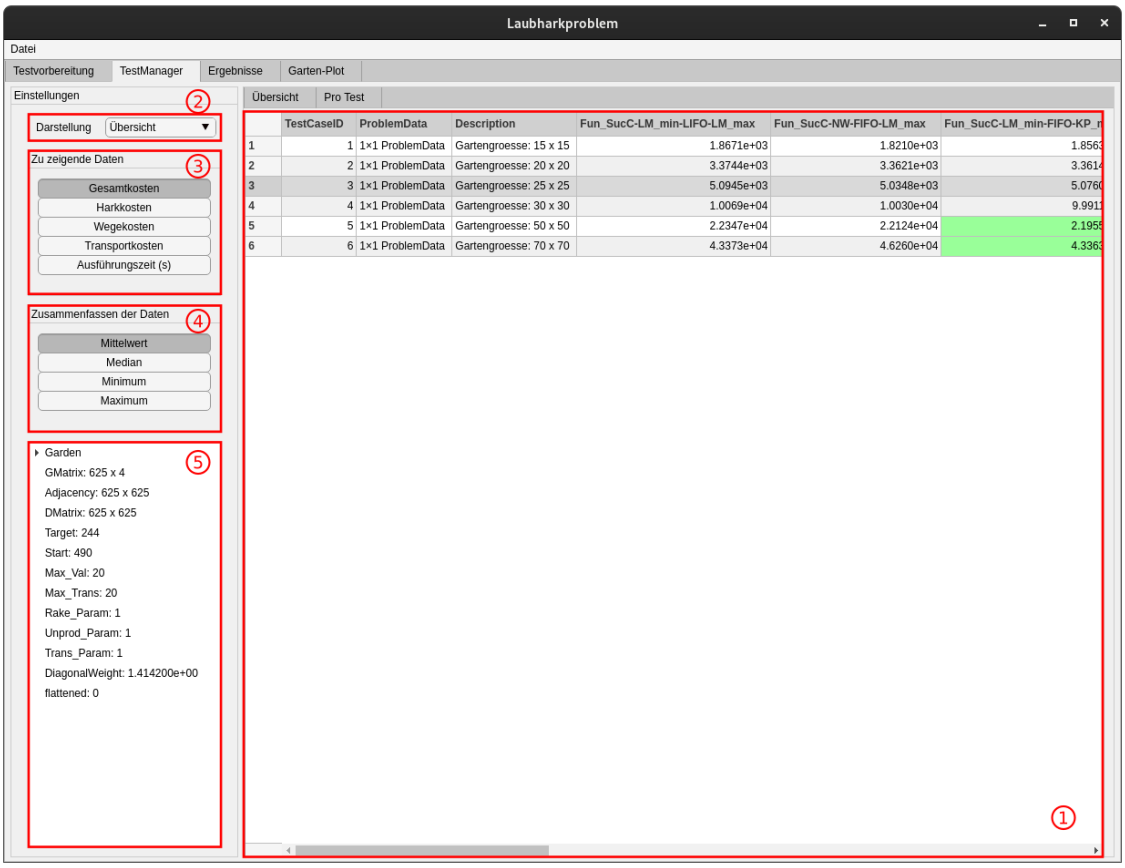


Abbildung 30: *Zweiter Tab des `TestManager`–GUI mit Darstellungsart „Übersicht“.*

- **Auswahl einer Zelle der ersten 3 Tabellenspalten**

  Die ersten drei Tabellenspalten enthalten keine darstellbaren Ergebnisse. Eine Auswahl einer einzelnen Zelle hier hat denselben Effekt wie die Auswahl einer ganzen Tabellenzeile (siehe nächster Eintrag).
- **Auswahl einer Tabellenzeile**

  Wird eine ganze Tabellenzeile ausgewählt (durch das Anklicken einer der fortlaufend durchnummerierten Zahlen ganz links im ersten Panel) wird automatisch das beste Ergebnis dieser Zeile ausgewählt und in den Tabs (3) und (4) grafisch dargestellt.
- **Auswahl mehrerer Zellen**

  Das Auswählen mehrerer Zellen hat keinen Effekt.
- **Auswahl einer Tabellenspalte**

  Das Auswählen einer Tabellenspalte hat keinen Effekt.

Hinweis

Um das Auffinden der besten Lösung pro durchgeführtem Test (also pro Tabellenzeile) zu erleichtern, werden diese Zellen in der Ansicht in Panel (1) mit grüner Hintergrundfarbe hervorgehoben. Auf diese Weise erlangt man schnell den Überblick, welche Algorithmen besonders gut abgeschnitten haben.

Hinweis

Ergebnisse für Tests, die noch nicht durchgeführt worden sind, werden in der tabellarischen Ansicht in Panel (1) mit `NaN` gekennzeichnet. Um konkrete Ergebnisse zu erhalten, müssen die Tests zunächst durchgeführt werden!

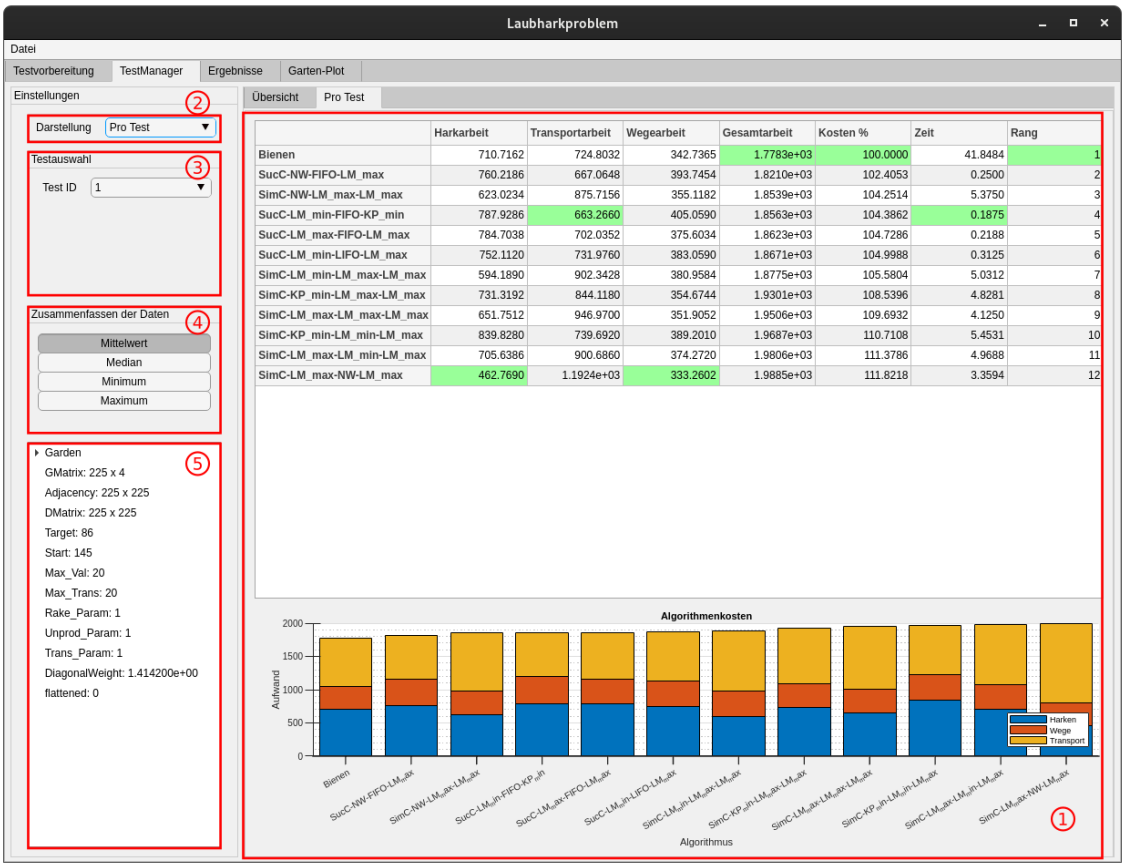
In Panel (3) kann der Nutzer auswählen, welcher Wert der Ergebnisse in Panel (1) dargestellt werden soll.

In Panel (4) (Abbildungen 31) wird ausgewählt, wie Ergebnisse in der tabellarischen Ansicht zusammengefasst werden sollen. Diese Einstellung betrifft ausschließlich die bionischen Algorithmen, da diese in einer Monte–Carlo–Simulation ausgeführt werden (siehe `StochasticWrapper`) und daher zu einem Testszenario mehrere Ergebnisse erzeugen. Da es nicht möglich ist, in dieser tabellarischen Ansicht mehrere Ergebnisse anzuzeigen, müssen die verschiedenen Werte zusammengefasst werden (zum Beispiel durch die Bildung eines Mittelwerts).

Darstellungsart „Pro Test“
Panel (1) zeigt die Ergebnisse in tabellarischer Form, wie sie die Methode `get_results()` des `TestManager` unter Verwendung des `„From"` Arguments zurückgibt. Wie in der „Übersicht“-Darstellung, führt die Auswahl einer Tabellenzelle zur Darstellung der Lösung für den ausgewählten Algorithmus in den „Ergebnisse“- und „Garten–Plot“-Tabs.
Während Panel (4) und (5) die gleiche Funktion erfüllen wie in der „Übersicht“ Darstellungsart, wird über die Schaltfläche in Panel (3) der darzustellende Test ausgewählt. Nach der Auswahl eines Tests wird die Ansicht in Panel (5) sofort aktualisiert.

5.2.3.4 Der „Ergebnisse“–Tab

Dieser Tab (Abbildung 32) dient der Visualisierung von Ergebnissen, die zuvor im zweiten Tab (Abbildung 30 und 31) ausgewählt worden sind.



	Harkarbeit	Transportarbeit	Wegearbeit	Gesamtarbeit	Kosten %	Zeit	Rang
Bienen	710.7162	724.8032	342.7365	1.7783e+03	100.0000	41.8484	1
SucC-NW-FIFO-LM_max	760.2186	667.0648	393.7454	1.8210e+03	102.4053	0.2500	2
SimC-NW-LM_max-LM_max	623.0234	875.7156	355.1182	1.8539e+03	104.2514	5.3750	3
SucC-LM_min-FIFO-KP_min	787.9286	663.2660	405.0590	1.8563e+03	104.3862	0.1875	4
SucC-LM_max-FIFO-LM_max	784.7038	702.0352	375.6034	1.8623e+03	104.7286	0.2188	5
SucC-LM_min-LIFO-LM_max	752.1120	731.9760	383.0590	1.8671e+03	104.9988	0.3125	6
SimC-LM_min-LM_max-LM_max	594.1890	902.3428	380.9584	1.8775e+03	105.5804	5.0312	7
SimC-KP_min-LM_max-LM_max	731.3192	844.1180	354.6744	1.9301e+03	108.5396	4.8281	8
SimC-LM_max-LM_max-LM_max	651.7512	946.9700	351.9052	1.9506e+03	109.6932	4.1250	9
SimC-KP_min-LM_min-LM_max	839.8280	739.6920	389.2010	1.9687e+03	110.7108	5.4531	10
SimC-LM_max-LM_min-LM_max	705.6386	900.6860	374.2720	1.9806e+03	111.3786	4.9688	11
SimC-LM_max-NW-LM_max	462.7690	1.1924e+03	333.2602	1.9885e+03	111.8218	3.3594	12

Abbildung 31: *Zweiter Tab des `TestManager`–GUI mit Darstellungsart „Pro Test“.*

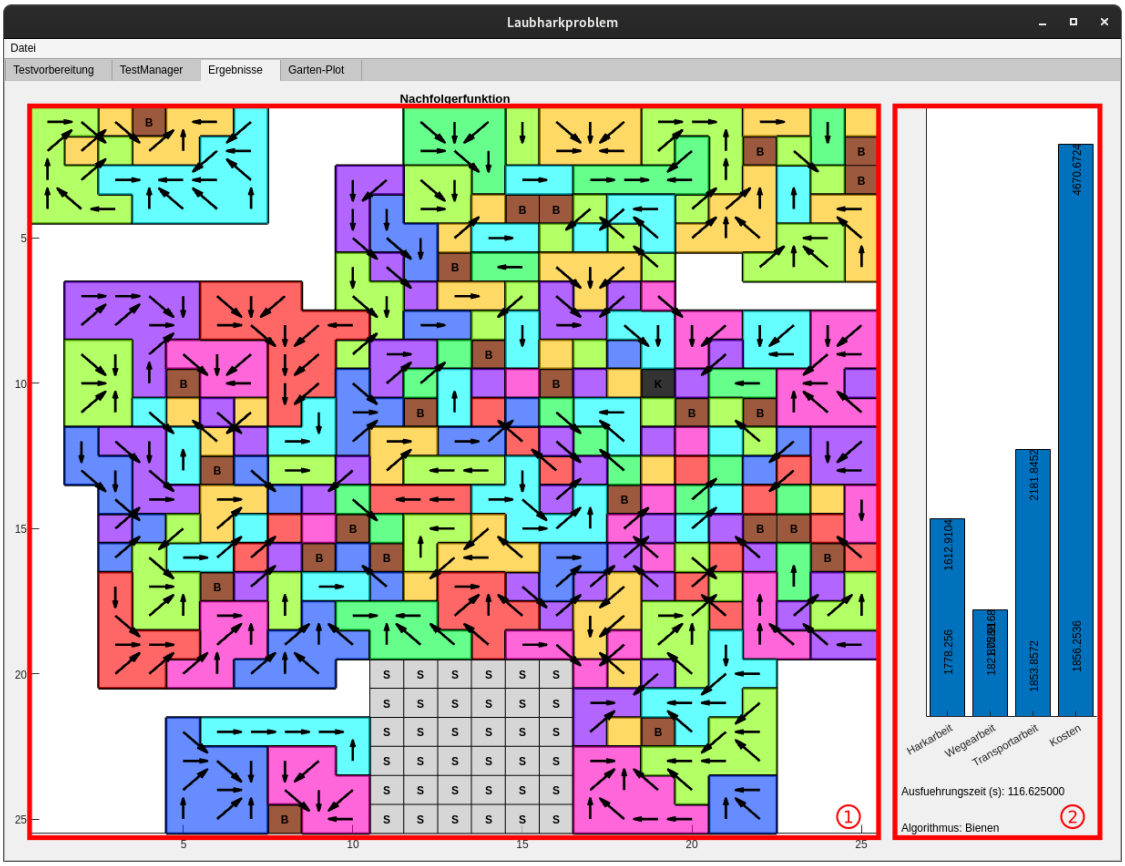


Abbildung 32: *Dritter Tab des `TestManager`–GUI.*

Panel (1) zeigt eine grafische Visualisierung der erzeugten Nachfolgerfunktion. In der Visualisierung haben die verschiedenen Zellen folgende Bedeutung:

- **Braune Zelle mit „B“**
  Diese Felder sind durch Bäume blockiert.
- **Graue Zelle mit „S“**
  Diese Felder sind durch einen Gartenschuppen blockiert.
- **Schwarze Zelle mit „K“**
  Hier ist der Kompost.

Alle anderen Farben verdeutlichen zusammen mit den schwarzen Umrandungen die Clusterzugehörigkeit der jeweiligen Zellen. Die Farben sind mit einer einfachen Heuristik so gewählt, dass benachbarte Cluster möglichst nicht dieselbe Farbe haben. Die Pfeile innerhalb der Cluster zeigen an, wohin das Laub geharkt wird. Laubquellen sind daran zu erkennen, dass von ihnen nur Pfeile weg-, aber nicht hinführen. Demgegenüber sind Laubsenken daran zu erkennen, dass nur Pfeile hin-, aber nicht wegführen.

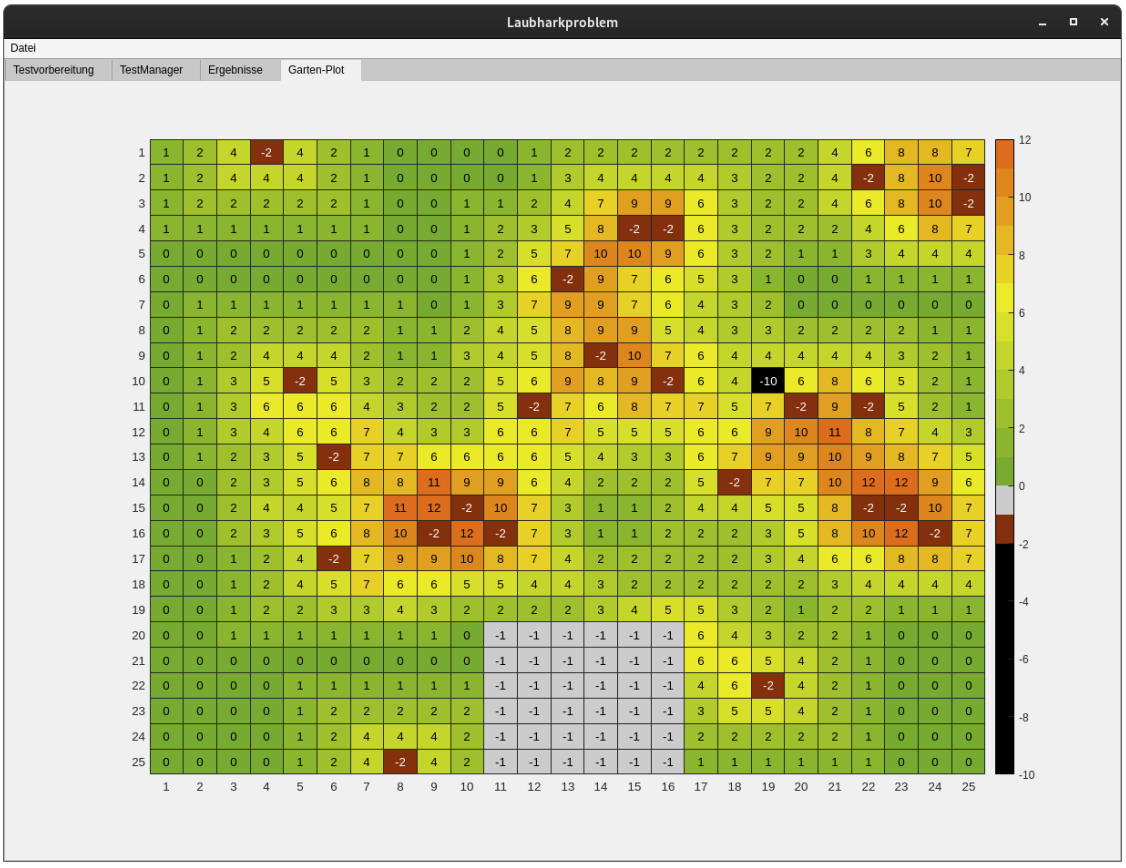


Abbildung 33: *Vierter Tab des `TestManager`–GUI.*

Alle nicht eingefärbten Felder (in der Darstellung „weiß“) werden bei der Lösung des Problems nicht berücksichtigt. Auf ihnen liegt während des Harkprozesses

zu keinem Zeitpunkt Laub, daher finden hier weder Harkvorgänge noch Abtransporte statt.

Panel (2) zeigt die Ergebniswerte für den ausgewählten Algorithmus an. Das Balkendiagramm zeigt die einzelnen Kostenbestandteile sowie die Gesamtkosten. Darunter ist die Ausführungszeit des Algorithmus angegeben sowie der Name des Algorithmus, wie er in der Tabelle angegeben ist.

5.2.3.5 Der „Garten–Plot“–Tab

Der vierte Tab (Abbildung 33) stellt den ausgewählten Garten grafisch dar. Er dient zur Vervollständigung der textuellen Informationen über `ProblemData`–Objekte (Testszenarien), die in den Tabs in Abbildung 29, 30 oder 31 ausgewählt worden sind.

5.3 Bezugsquelle des MATLAB-Programms `lhp`

Das Programm `lhp` mit Beispielen sowie der Dokumentation kann von unserem GitHub-Repository https://github.com/AMIT-HSBI/LHP heruntergeladen werden.

6 Vergleichstests und Auswertungen

Das im vorherigen Kapitel vorgestellte MATLAB-Programm `lhp` zielt in erster Linie darauf ab, die darin implementierten Verfahren hinsichtlich der Lösungsgüte[64], des Rechenaufwandes und des effizienten Einsatzes zur Lösung von Laubharkproblemen vergleichend gegenüberzustellen. Dabei stehen auch die bionischen Verfahren im Fokus der Betrachtungen, inwieweit sie zur Optimierung von ähnlich gelagerten Ver- und Entsorgungsprozessen geeignet sind.

Für eine umfassende Untersuchung bedarf es einer Aufstellung aller denkbaren Gärten im Hinblick auf Größe, Form, innere Struktur[65] und Laubverteilung. Eine derart umfangreiche Datengrundlage liegt den hier beschriebenen Untersuchungen nicht zugrunde, vielmehr liegt eine Zusammenfassung von verschiedenen Beispielgärten vor, die den Autoren als eine hinreichend repräsentative Vielzahl erschien, um zumindest Trendaussagen zu den untersuchten Fragestellungen treffen zu können (Näheres hierzu siehe Abschnitt 6.3). Auch wird aus der Vielzahl interessanter Fragestellungen, die in diesem Zusammenhang untersucht werden könnten[66], hier eine Beschränkung auf eine Auswahl an Untersuchungsgegenständen vorgenommen (Näheres hierzu siehe Abschnitt 6.1). Die Beantwortung dieser untersuchten Fragestellungen erfolgt in Abschnitt 6.4.

6.1 Untersuchte Fragestellungen

Bei der vergleichenden Gegenüberstellung der Lösungsverfahren stellt sich unmittelbar die Frage nach den maßgeblichen Vergleichsgrößen. In der vorliegenden Problemstellung ergibt sich der beim Laubentsorgungsprozess auftretende, zu minimierende Entsorgungsaufwand als das primäre Gütemaß und gibt die Lösungsgüte des Verfahrens an. Darüber hinaus können, je nach individuellem

[64] Die Lösungsgüte eines Verfahrens wird allgemein an dem durch das Verfahren ermittelten Zielfunktionswert eines zugrundeliegenden Optimierungsproblems gemessen, und zwar im Vergleich zu den Ergebnissen durch alternative Lösungsverfahren.

[65] Bei der inneren Struktur der Gärten ist insbesondere die *Dichte* von Interesse, d.h. der Anteil von blockierten Feldern zur Anzahl aller Felder eines Gartens.

[66] Zu offenen Fragestellungen siehe auch das abschließende Kapitel 7.

Realproblem, auch einzelne Teilaufwände wie Hark-, Transport- oder Wegeaufwand besonders entscheidungsrelevant sein. Da diese Teilaufwände allerdings über die Modellparameter α_H, α_T bzw. α_W gezielt angesteuert werden können, bleibt der Gesamtaufwand die zentrale Messgröße für die Lösungsgüte.

Darüber hinaus ist allerdings auch der jeweilige Rechenaufwand (Rechenzeit) als sekundäres Gütemaß von Interesse. Diese beiden Größen (Gesamtaufwand, Rechenzeit) treten in der Regel als konkurrierende Messgrößen auf, d.h. für eine Verringerung des Zielfunktionswertes wird in der Regel mehr Rechenzeit benötigt. Dabei stellt sich beispielsweise die Frage, für eine wie geringe Verbesserung des Zielfunktionswertes ein wie großer Rechenaufwand „lohnenswert“ ist. Diese Fragestellung führt unweigerlich zum Begriff der *Effizienz* und beim Verfahrensvergleich nach dem jeweiligen *Effizienzwert* der einzelnen Verfahren, wodurch dem Entscheidungsträger die Möglichkeit an die Hand gegeben wird, die subjektive Wichtigkeit der beiden Gütemaße *Gesamtaufwand* und *Rechenzeit* in die Bewertung einfließen zu lassen. Eine Möglichkeit der Effizienzbewertung der Verfahren als übergeordnete Vergleichsgröße wird im Anhang A.1 vorgeschlagen.

Zum Gütevergleich insbesondere der deterministischen Verfahren ergeben sich viele Einzelfragen. Von vorrangiger Bedeutung ist die Frage, ob es hinsichtlich der Lösungsgüte dominierende Heuristiken unter den implementierten Verfahrensvarianten gibt. Damit sind solche Varianten gemeint, die bei vielen oder sogar nahezu allen Testgärten zu den jeweils besten Lösungen geführt haben und sich dadurch als besonders empfehlenswert herausstellen. Aber auch solche Varianten, die nur sehr vereinzelt unter den besten Lösungen auftreten, könnten dennoch von Interesse sein, sei es, dass sie bei speziell strukturierten bzw. laubverteilten Beispielgärten besonders gute Lösungen generieren, oder sei es auch nur zur Erzeugung von alternativen Startlösungen für die bionischen Verfahren. Allerdings stellt sich als das wesentliche Untersuchungsziel, eine beschränkte (möglichst kleine) Auswahl unter den deterministischen Heuristiken zu ermitteln, die bzgl. der gesamten Testreihe die jeweils beste Lösung „überdeckt“, um dieses „Best-Mix“ als Grundlage für weitere Untersuchungen und Tests empfehlen zu können.[67]

Darüber hinaus sind die jeweiligen Rechenzeiten der Verfahren von Interesse. Dabei stellt sich die grundsätzliche Frage nach dem tendenziellen Verhalten der Rechenzeit zur Gartengröße. Insbesondere aber dürften die Rechenzeiten der deterministischen Verfahren beim Einsatz der Laubmengenobergrenzenvariation

[67] Es sei am Rande erwähnt, dass hierzu ein spezielles kombinatorisches Optimierungsproblem aus der Klasse der *Überdeckungsprobleme* (*Covering problems*) zu lösen ist.

an Bedeutung gewinnen, da für jede betroffene Heuristik ein vielfacher Durchlauf vorgenommen wird (vgl. Abschnitt 3.4). Es stellt sich somit die Frage, ob es grundsätzlich lohnenswert ist, die zusätzliche Rechenzeit für eine Laubmengenobergrenzenvariation im Hinblick auf die Verbesserung der Lösungsgüte zu investieren oder ob dieser Zusatzaufwand nur bei bestimmten Verfahrensvarianten oder bei bestimmten Gartenstrukturen empfehlenswert ist.

Schließlich stellt sich auch noch die Frage, inwieweit das Ranking, welches sich beim Gütevergleich der implementierten Heuristiken ergibt, von verschiedenen Einflussgrößen wie Gartengröße und -struktur oder Laubmengenverteilung variiert. Gibt es beispielsweise „durchweg dominierende“ Verfahrensvarianten, die sich unabhängig von solchen Einflussgrößen als besonders empfehlenswert herauskristallisieren oder muss beim Ranking spezifiziert bzw. klassifiziert werden (z.B. kleine, mittlere und große Gärten).

Von besonderem Interesse ist auch die Einbeziehung der bionischen Lösungsmethoden in den Gütevergleich.[68] Dabei interessiert sowohl das Abschneiden der beiden bionischen Lösungsmethoden im direkten Vergleich als auch im Vergleich mit den deterministischen Heuristiken. Letzteres soll zunächst auf der Grundlage einer jeweiligen „Stand-alone-Anwendung“ untersucht werden, d.h. dass die unterschiedlichen Lösungsansätze komplett unabhängig voneinander agieren. Der Fall, dass die bionischen Verfahren als Verbesserungsverfahren der deterministischen Heuristiken fungieren, wobei letztere die Rolle von Eröffnungsverfahren übernehmen, wird in der folgenden Frage problematisiert.

Auf dem Weg zur Suche nach besseren Lösungen benötigen bionische Verfahren prinzipiell die Vorgabe von Startlösungen[69]. Diese können auf verschiedene Weise ermittelt worden sein, indem im Extremfall triviale Lösungen (z.B. gar nicht Harken) oder aber bereits durch andere Verfahren ermittelte Harkstrategien als Anfangslösungen genommen werden. Je nach den Suchprinzipien der bionischen Verfahren können verschiedene Startlösungen unterschiedlichen Einfluss auf die Güte der Suchergebnisse haben. Ein positiver Effekt könnte eine deutliche Verringerung der Rechenzeiten sein, da bei Vorgabe von bereits guten Startlösungen viele Schritte im Suchprozess der Verfahren eingespart werden könnten. Allerdings kann sich (theoretisch) dabei auch ein negativer Effekt einstellen, denn durch eine strukturierte Vorgabe von Startlösungen kann die Suchrichtung unvor-

[68] An dieser Stelle sei (nochmals) darauf hingewiesen, dass von dem genetischen oder dem Bienen-Algorithmus nicht gesprochen werden kann, da es sich um eine Vielzahl von Ausprägungen aus der jeweiligen Klasse dieser Metaheuristiken handelt. Vielmehr sind im Folgenden stets die Untersuchungsergebnisse zu den im Programm `lhp` implementierten Algorithmen gemeint.

[69] Speziell bei den genetischen Verfahren spricht man von *Startpopulationen*.

teilhaft eingeschränkt werden und eine unabhängige Suche nach möglicherweise besseren Lösungen in anderen Gebieten des gesamten Lösungsspektrums verhindern.

Daher soll untersucht werden, ob ein signifikanter Einfluss einer gezielten Vorgabe von Startlösungen auf die Lösungsgüte der implementierten bionischen Verfahren besteht und wie sensitiv bzw. robust die Verfahren sowohl hinsichtlich der Lösungsgüte als auch bzgl. der Rechenzeit darauf reagieren.

Die einzelnen Untersuchungsergebnisse dazu befinden sich in Abschnitt 6.4. Weitere interessante und daher untersuchungswürdige Fragestellungen, die in dieser Abhandlung nicht weiter verfolgt werden, sind im Ausblick (Kapitel 7) zu finden.

6.2 Gütekriterien

Um die Lösungsgüte der Verfahrensvarianten gesamtheitlich über verschiedene Tests messbar zu machen, lassen sich eine Reihe von Gütekriterien anwenden, wobei zunächst auf den Bewertungszweck zu achten ist, d.h. soll eine Einzelbewertung der Verfahren erfolgen (Abschnitt 6.2.1) oder soll eine möglichst geeignete Verfahrensauswahl bestimmt werden (Abschnitt 6.2.2). Dabei ist jeweils auch noch darauf zu achten, ob sich die Gütebewertung allein auf die primäre Ergebnisgröße (Gesamtaufwand) bezieht oder gleichzeitig auch die sekundäre Ergebnisgröße (Rechenaufwand) einbezogen wird (bikriterielle Betrachtung). Bikriterielle Vergleichsgrößen werden in Abschnitt 6.2.3 vorgestellt, während sich die Gütekriterien in Abschnitt 6.2.1 und 6.2.2 allein auf die Ergebnisgröße *Gesamtaufwand* beziehen.

6.2.1 Gütekriterien zur Einzelbewertung

Bezogen auf die Ergebnisse einer Testreihe bieten sich folgende Einzelbewertungen der Verfahren an:

- **Häufigkeit 1. Platz**:
  Es wird gezählt, wie oft ein Verfahren den besten Ergebniswert (niedrigster Gesamtaufwand) erzielt hat.
- **Häufigkeit TOP x**:
  Anhand eines Platzrankings wird gezählt, wie oft der Ergebniswert eines

Verfahrens unter die x besten Werte fällt.[70] Dabei ist die Anzahl x sinnvoll festzulegen, z.B. TOP 5 oder TOP 10.[71]

- **Durchschnittliche Platzierung**:
  Anhand eines Platzrankings wird die durchschnittliche Platzierung eines Verfahrens ermittelt.[72]
- **Worst-Case-Platzierung**:
  Anhand eines Platzrankings wird die schlechteste Platzierung eines Verfahrens ermittelt.

Die bisherigen Gütekriterien orientieren sich an an einer ordinalen Messgröße (Platzierung von Ergebniswerten), bei denen die Abstände der Ergebniswerte nicht einbezogen werden, im Gegensatz zu folgenden Gütekriterien:

- **Durchschnittliche prozentuale Abweichung vom besten Ergebniswert**:
  Hier wird zunächst für jeden Garten aus der Testreihe der beste Ergebniswert (über alle Verfahren) bestimmt, um dann den Abstand des Ergebniswertes eines jeden Verfahrens zu diesem Bestwert zu ermitteln. Zwecks Vergleichbarkeit für verschiedene Testgärten werden die prozentualen Abstände genommen, wobei der jeweilige Bestwert als Basiswert die 0%-Marke bildet, um daraus über alle Tests die Durchschnittsabweichung [%] für jedes Verfahren bilden zu können.
- **Maximale prozentuale Abweichung vom besten Ergebniswert**:
  Zu jedem Verfahren wird die maximale prozentuale Abweichung vom jeweiligen Bestwert der einzelnen Tests ermittelt.

Diese beiden Gütekriterien orientieren sich an der kardinalen Messgröße *Gesamtaufwand* und dienen als „Verlässlichkeitsmaße“, indem sie darüber Auskunft geben, mit welcher Abweichung vom Bestwert im Mittel bzw. schlimmstenfalls zu rechnen ist.

6.2.2 Gütekriterien zur Auswahlbildung

Wenn es darum geht, aus der Vielzahl der implementierten Verfahrensvarianten eine empfehlenswerte Auswahl zu treffen, welche für beliebige Gärten gute Lösungsergebnisse erwarten lassen, bietet sich die Ermittlung einer *kompletten*

[70] Alternativ könnte auch gezählt werden, wie viele Verfahren zu einem bestimmten Testszenario bessere Ergebnisse erzielt haben, d.h. dass bei übereinstimmenden Zielwerten (identischer Gesamtaufwand) entsprechend viele nachfolgende Plätze nicht vergeben werden (Beispiel: Bei Verleihung von zwei Silbermedaillen entfällt die Vergabe der Bronzemedaille.). In den Testergebnissen in Abschnitt 6.4 erfolgt die Rankingbildung nach dem oben vorgestellten Prinzip.

[71] Man beachte, dass dieses Kriterium eine Verallgemeinerung des ersten bzw. das erste ein Spezialfall des zweiten darstellt (x = 1).

[72] Da es sich bei der Kennzahl *Platzierung* um eine ordinale Messgröße handelt, wird der jeweilige Zentralwert ermittelt.

Überdeckungsmenge an. Damit ist eine Verfahrensauswahl gemeint, welche garantiert, dass zu jedem Garten einer Testreihe der jeweils beste Ergebniswert, der von allen Verfahren erreicht wird, bereits „aus der Mitte“ der Verfahrensauswahl erzielt wird, d.h. dass zu jedem besten Ergebniswert mindestens eines der ausgewählten Verfahren diesen Bestwert erreicht. Mit dem Ziel, die Anzahl der ausgewählten Verfahren möglichst gering zu halten, sollen diese Überdeckungsmengen *irreduzibel* sein, d.h. dass durch das Entfernen eines beliebigen Verfahrens aus dieser Auswahlmenge die Überdeckungseigenschaft verloren geht.[73]

Für die Ermittlung einer kompletten Überdeckungsmenge bietet sich ein recht einfaches Greedy-Verfahren an: Anhand einer 0-1-Matrix, bei der die Zeilen den Tests und die Spalten den Verfahren entsprechen und in der die Komponente (i,j) den Wert 1 hat, wenn das Verfahren j beim Test i den besten Ergebniswert erzielt (0 sonst), entsprechen die Spaltensummen den Häufigkeiten des 1. Platzes. Es wird das Verfahren mit größtem Häufigkeitswert zur Auswahlmenge gezählt und alle Tests, die von diesem Verfahren mit 1 „überdeckt“ werden, gestrichen (z.B. durch Nullsetzung der zugehörigen Zeilen). Diese Prozedur wird mit der geänderten Matrix solange wiederholt, bis alle Tests überdeckt bzw. gestrichen sind.

Hieraus wird das folgende binäre Gütekriterium abgeleitet:

- **Zugehörigkeit zur kompletten Überdeckungsmenge**:
  Es werden alle Verfahren, die zur ermittelten kompletten Überdeckungsmenge gehören, für eine Auswahl empfohlen.

Dieses Auswahlkriterium kann auch dahingehend verallgemeinert werden, dass anstelle einer kompletten Überdeckungsmenge (100%) eine *beschränkte Überdeckungsmenge* bestimmt wird, welche nicht mehr alle Tests, sondern nur noch eine vorgegebene Prozentzahl der Tests überdeckt. Darüber hinaus kann die Forderung der Überdeckung des 1. Platzes auch dahingehend verallgemeinert werden, dass nur noch eine jeweilige Mindestplatzierung unter den Top x verlangt wird (zu vorgegebenem x). Beide Verallgemeinerungen tragen dazu bei, die Anzahl der ausgewählten Verfahren unter Inkaufnahme einer bewusst zugelassenen Qualitätsminderung zu verringern. Die Änderungen lassen sich sehr einfach in die obige Greedy-Heuristik integrieren.

[73] Zwar zielt die Ermittlung einer kompletten Überdeckungsmenge aufgrund der Eigenschaft, irreduzibel zu sein, per se darauf ab, sich bei der Auswahl auf möglichst wenige Verfahren zu beschränken, garantiert aber nicht, dass es sich um eine anzahlmäßig minimale Überdeckungsmenge handelt. Das Problem der Bestimmung von minimalen Überdeckungen gehört bekanntlich zur Klasse der harten kombinatorischen Optimierungsprobleme.

Dieses Auswahlkriterium ist insbesondere dann empfehlenswert, wenn die komplette bzw. beschränkte Überdeckungsmenge zuvor anhand einer repräsentativen Testreihe mit vielen und unterschiedlichen Testgärten ermittelt worden ist, sodass mit hoher Wahrscheinlichkeit davon ausgegangen werden kann, dass sich auch für andere Gärten eine gute Lösung „aus der Mitte“ dieser Verfahrensauswahl ergeben wird.

6.2.3 Bikriterielle Vergleichsgrößen

Alle bisher vorgestellten Gütekriterien beziehen sich allein auf die primäre Ergebnisgröße *Gesamtaufwand*. Um auch die sekundäre Ergebnisgröße *Rechenaufwand* in die Gütebewertung einzubeziehen, kann auf Effizienzwerte zurückgegriffen werden (hierzu siehe Anhang A.1).

- **Häufigkeit bester Effizienzwert**:
  In Analogie zum obigen Gütekriterium *Häufigkeit 1. Platz* (bzgl. Gesamtaufwand) wird zu jedem Verfahren die Häufigkeit des besten Effizienzwertes gezählt.
- **Häufigkeit TOP x**:
  In Analogie zum obigen Gütekriterium *Häufigkeit TOP x* (bzgl. Gesamtaufwand) wird zu jedem Verfahren die Häufigkeit der jeweiligen Platzierung unter den x besten Effizienzwerten gezählt.

Es sei hier angemerkt, dass zur Bestimmung der Effizientwerte eine konkrete Vorgabe des Entscheidungsträgers zu seiner individuellen Gewichtung der beiden Zielgrößen *Gesamtaufwand* und *Rechenaufwand* vorliegen muss.[74]

Das folgende Gütekriterium basiert ebenfalls auf der bikriteriellen Betrachtung der Ergebnisgrößen *Gesamtaufwand* (GA) und *Rechenaufwand* (RA); es ist allerdings unabhängig von deren Gewichtung, indem alle Fälle von möglichen Gewichtungen Berücksichtigung finden. Vorbereitend werden zu jedem Test die Pareto-optimalen Ergebnispaare (GA;RA) bestimmt, wobei die zugehörigen Verfahren die jeweilige „Pareto-Auswahlmenge“ bildet .[75] Hieraus wird das folgende Gütekriterium abgeleitet:

- **Häufigkeit der Zugehörigkeit zu einer Pareto-Auswahlmenge**:
  Es wird zu jedem Verfahren die Häufigkeit der Zugehörigkeit zu den Pareto-Auswahlmengen über alle Tests ermittelt.

Es ist durchaus sinnvoll, diese Häufigkeitszählung durch die Vorgabe einer maximalen Verschlechterung gegenüber dem besten Zielfunktionswert (Gesamtauf-

[74] Hierzu siehe den Vorschlag in Anhang A.1.
[75] Hierzu siehe die Ausführungen in Anhang A.1.

wand) und/oder durch die Vorgabe einer maximalen Rechenzeit einzuschränken.

Die vorgestellten Gütekriterien[76] lassen sich in vielfältiger Weise zur Bestimmung einer zu empfehlenden Verfahrensauswahl einsetzen, indem (individuelle) Erfüllungsgrade für alle oder einige Kriterien vorgegeben werden und nur diejenigen Verfahren zur Verfahrensauswahl gezählt werden, die alle gesetzten Restriktionen erfüllen.

6.3 Testvoraussetzungen

Die ausgewählten Testgärten unterscheiden sich in Größe, Form, Dichte und Laubverteilung. Die Gartengröße entspricht der Anzahl aller Felder des Gartens, welche sich auf kanonische Weise aus Zeilen- und Spaltenzahl der $m \times n$-Matrix des zugehörigen Gartenmodells ergibt (vgl. Abschnitt 2.1). Zu den „kleinen“ Gärten zählen solche mit bis zu 1000 Feldern (z.B. 25×25), zu den „mittleren“ solche mit bis zu 4000 Feldern (z.B. 50×50), zu den „großen“ solche mit mehr als 4000 Feldern (z.B. 100×100). Die Gartenform ergibt sich aus dem Verhältnis von Zeilen- und Spaltenzahl m und n der Modellmatrix (quadratisch, rechteckig).

Die Testgärten wurden mit dem MATLAB-Skript ‘`gen_gardens.m`‘ erzeugt, welches als Teil des LHP-Projektes unter ‘`lhp.benchmark.resources`‘ zu finden ist. Die generierten Gärten werden in mehreren ‘`.mat`‘-Dateien gespeichert, die wiederum von der ‘`ProblemGenerator`‘-Klasse eingelesen und in ‘`ProblemData`‘-Objekte umgewandelt werden. Dabei werden die Gärten mit verschiedenen Laubmustern und entsprechenden Bäumen befüllt.

Die Funktion zum Erzeugen der Testgärten ist so geschrieben, dass anhand unterschiedlicher „Variationsprinzipen“ zwei verschiedene Gartentypen erzeugt werden:

Typ I: Formfixe Größenvariation
: Diese Gärten haben eine quadratische Grundfläche. Insgesamt wurden 7 verschiedene „Basisgärten“ erzeugt, deren Seitenlängen je 13, 18, 25, 35, 50, 70 und 100 [Feldlängeneinheiten] betragen. Diese Werte wurden derart gewählt, dass die Gesamtzahl der Felder pro Garten mit jeder Größenkategorie ungefähr um den Faktor 2 steigt.

Typ II: Größenfixe Formvariation
: Diese Gärten entstehen nach einem anderen Aufbauprinzip: Hierbei wurde die Gesamtzahl der Felder auf 1225 festgelegt, und jeweils die Länge ei-

[76] Hiermit sind alle Kriterien aus den Abschnitten 6.2.1 - 6.2.3 gemeint.

ner Seite verkleinert, um rechteckige Gärten zu erzeugen. Die insgesamt 4 „Basisgärten“ haben die Maße 35×35, 25×49, 7×175 und 5×245.

Innerhalb der Gärten kommen drei verschiedene Laubmuster zum Einsatz:

- Gleichverteilte Laubmenge aus Interval [1; 5]
- Gleichverteilte Laubmenge aus Interval [1; 10]
- Pyramidenförmiges Laubmuster, zentriert um die „Baumstämme“

Ein weiteres Unterscheidungsmerkmal der Gärten ist neben dem Laubmuster, das der ‘`ProblemGenerator`‘ erzeugt, die sog. „Baumdichte“. Hierbei handelt es sich um das Verhältnis von Anzahl der durch Bäume blockierte Felder zur Gesamtzahl aller Gartenfelder. Ein Vorversuch hat ergeben, dass eine Baumdichte oberhalb von 10 % keine brauchbaren Gärten erzeugt, da hier kein zusätzlicher Informationsgewinn über die Güte der Algorithmen erlangt wird. Daher wurden die Tests auf Gärten mit einer Baumdichte von 5 % und 10 % beschränkt.

Diese Garteneinteilung ist in Tabelle 10 zusammengefasst. Es ergeben sich insgesamt 66 verschiedene Testszenarien (und zwar 42 vom Typ I und 24 vom Typ II).[77]

Tabelle 10: *Zusammenfassung der Testgärten*

Typ	quadratisch		rechteckig
Typ I	klein	13×13 18×18 25×25	
	mittel	35×35 50×50	
	groß	70×70 100×100	
Typ II	mittel	35×35	25×49 7×175 5×245

⇒

Dichte
0,05
0,10

⇒

Laubverteilung
baumzentriert
gleichverteilt [1;5]
gleichverteilt [1;10]

Die den folgenden Auswertungen zugrundeliegenden Tests wurden auf Rechnern des Studiengangs Angewandte Mathematik der Hochschule Bielefeld durchgeführt. Die Testdurchführungen fanden auf DELL OptiPlex 7070 Rechnern statt. Diese verfügen über einem Intel© Core™ i7-9700 CPU @ 3.00GHz Prozessor mit 8 Kernen. Zusätzlich sind die Rechner mit 32 GB DDR4 RAM ausgestattet. Wegen des Aufwands der Berechnungen wurden diese auf 7 Rechner (für je ein

[77] Die Testgärten der Größe 35x35 vom Typ I und II sind nicht identisch.

explizites Hubbestimmungsverfahren) verteilt.

Bei den Tests wurden drei verschiedene Parametereinstellungen der Aufwandsfaktoren betrachtet:

$(P1)$ $\alpha_H = \alpha_T = \alpha_W = 1$,
$(P2)$ $\alpha_H = 1, \alpha_T = 2, \alpha_W = 0,5$,
$(P3)$ $\alpha_H = 1, \alpha_T = 3, \alpha_W = 0,25$.

Zusätzlich waren folgende Modellparameter fix eingestellt:

- Maximale Laubmenge eines Feldes: $\overline{M} = 25$,
- Maximales Ladevolumen: $\overline{T} = 25$,
- Diagonalwert: $\sqrt{2}$.

Im MATLAB-Programm `lhp` sind aktuell 980 verschiedene deterministische Verfahrensvarianten implementiert. Diese lassen sich anhand der Grundprinzipien der Clusterbildung (sukzessiv und simultan), die in Kapitel 3 ausführlich beschrieben sind, in die beiden Grundverfahren *SucC* und *SimC* unterteilen. Zudem ist die einfache Zickzack-Strategie implementiert. Die beiden Grundverfahren liefern anhand von möglichen Auswahlkriterien verschiedene Verfahrensvarianten. Die *SucC*-Varianten unterscheiden sich durch die Auswahlkriterien für Repräsentanten und Aufnahmekandidaten sowie durch die Aufnahmereihenfolge (LIFO, FIFO); vgl. Abschnitt 3.1. Die *SimC*-Varianten ergeben sich durch die Auswahlkriterien für Zuweisungs- und Kontaktkandidaten; vgl. Abschnitt 3.2. Während bei den *SucC*-Varianten bereits durch die Repräsentantenwahl eine gezielte Hubbestimmung vorgenommen wird, erfolgt eine solche bei den *SimC*-Varianten eher beiläufig, weswegen drei Methoden für die Hubbestimmung direkt schon innerhalb des Grundverfahrens wählbar sind, ohne dass das zusätzliche Modul der expliziten Hubbestimmung aufgerufen werden muss. Falls keine verfahrensinterne Hubbestimmung erfolgt, bekommt die Variante die Bezeichnung *Impl*. Demnach gibt es 18 Varianten vom Grundtyp *SucC* und 64 Varianten vom Grundtyp *SimC*. Zusammen mit dem Zickzack-Verfahren ergeben sich 83 Verfahrensvarianten. Einen Überblick liefert die Tabelle 11.

An alle Verfahrensvarianten lassen sich verschiedene Module anhängen, um die Möglichkeit zu schaffen, verbesserte Lösungen zu generieren.[78] Mit Hilfe der *expliziten Hubbestimmung* (vgl. Abschnitt 3.3) werden alle vorläufigen Hubs ignoriert und entsprechend dem Auswahlkriterium endgültige Hubs für die Cluster gebildet, wobei 7 Optionen zur Verfügung stehen (inkl. ohne explizite Hubbestimmung bzw. einer optimalen Hubausrichtung unter Beibehaltung der Hubs[79]).

[78] Die Bezeichnung „Verbesserungsmodule“ trifft insofern nur begrenzt zu, da vereinzelt auch Verschlechterungen auftreten können.
[79] Zum Begriff der Hubausrichtung siehe Abschnitt 3.3, Seite 38.

Tabelle 11: *Zusammenfassung der deterministischen Verfahrensvarianten im Programm* `lhp`

Auswahlkriterien			
Wahlkriterium	Repräsentant	Breiten- oder Tiefensuche	Aufnahme-kandidat
Sukzessive Clusterbildung ($SucC$)	NW $LMmax$ $LMmin$	$FIFO$ $LIFO$	NW $LMmax$ $KPmin$
Wahlkriterium	Zuweisungs-kandidat	Kontakt-kandidat	Hub-auswahl
Simultane Clusterbildung ($SimC$)	NW $LMmax$ $LMmin$ $KPmin$	NW $LMmax$ $LMmin$ $KPmin$	$Impl$ $LMmax$ $LMmin$ $KPmin$
Zickzack			

$\Rightarrow$

Explizite Hub-bestimmung
$ohne$
NW
$LMmax$
$LMmin$
$KPmin$
$Median$
$Hubausrichtung$

$\Rightarrow$

Aussparung Leerfelder
$ohne$
mit

$\Rightarrow$

Laubmengen-obergrenzen-variation
$ohne$
mit

Durch das Aufrufen des Moduls zur *Aussparung von Leerfeldern* (*REN_1*) werden alle leeren Felder (d.h. Felder a mit der initialen Laubmenge $M(a) = 0$), die auch während des gesamten Harkprozesses laubleer bleiben, dem bisher zugehörigen Cluster entzogen und als einelementige Cluster isoliert, wodurch sie bei der Aufwandsberechnung komplett ignoriert werden. Andernfalls wird der Verfahrensvariante die Bezeichnung *REN_0* beigefügt. Schließlich ist das Einschalten einer *Laubmengenobergrenzenvariation* möglich: *OML_1*, andernfalls *OML_0* (vgl. Abschnitt 3.4).[80]

Anhand dieser Zusatzmodule lassen sich die 83 Grundvarianten durch 28 Modifikationsmöglichkeiten variieren, woraus sich insgesamt 2324 Varianten ergeben. Durch die mehrfache Wahlmöglichkeit der Hubbestimmung bei den *SimC*-Varianten entstehen redundante Verfahrensvarianten. Werden also beim Grundtyp *SimC* nur die 16 Auswahlmöglichkeiten zur Clusterbildung (ohne implizite Hubbestimmung) gezählt, vermindert sich die Anzahl der auszuwählenden Verfahrensvariationen auf 980.

Die einzelnen Verfahren erhalten anhand der Zugehörigkeit zu den Grundtypen und den verschiedenen Wahlmöglichkeiten entsprechende Bezeichnungen, z.B.

- *SucC-NW-FIFO-LM_max-OML_0-REN_1-PO_Median*
- *SimC-NW-FIFO-LM_max-OML_1-REN_1-PO_MinKompost*
- *Zickzack-OML_0-REN_0-PO_None*

Eine vollständige Liste aller 980 deterministischen Heuristiken mit den zugehörigen Kennziffern (ID) befindet sich im Anhang A.3. Dort findet sich auch eine Zusammenstellung der Kürzel, die im MATLAB-Programm `lhp` zur Namensgebung der Verfahrensvarianten benutzt werden, sowie deren Bedeutung bzw. Entsprechung (Seite 158).

Darüber hinaus werden die beiden in Kapitel 4 vorgestellten bionischen Lösungsmethoden in die Testrechnung und -auswertung einbezogen. Diese kamen bei den durchgeführten Testläufen mit folgenden speziellen Parametereinstellungen zum Einsatz:

Beim genetischen Algorithmus wurde als Standardparameter für die Populationsgröße 25 mit einer Reinitialisierungswahrscheinlichkeit von 4% festgelegt. Die maximale Iterationsanzahl beträgt eine Million Schritte, wobei der Stagnations-

[80] Die Abkürzungen *OML* und *REN* für die beiden Zusatzmodule stammen aus dem Programmcode und haben sich mittlerweile „verselbständigt“. *OML* steht für „Optimierung Max_Laub“, wobei *Max_Laub* der Name des Parameters in *ProblemData* ist (hierzu siehe Abschnitt 5.1); REN steht für „Remove Empty Nodes“.

wert zum Abbruch bei Iteration ohne Verbesserung bei 59 Schritten liegt. Die Mutationswahrscheinlichkeit wurde auf 72% und die Kreuzungswahrscheinlichkeit auf 64% festgelegt. Als Selektionsmethode wurde sich für die Rouletteradselektion mit einem Zeiger (R1) entschieden.

Der Bienenalgorithmus wurde im Fall von zufälligen Startlösungen abhängig von der Gartengröße (anz = Anzahl Felder) parametrisiert. Die Parameter waren $ns = \frac{3}{4} \cdot \sqrt{anz}, nb = ns, ne = \frac{1}{3} \cdot ns, nrb = 10, nre = 20$. Bei der Variante als Verbesserungsverfahren von bereits guten heuristischen Startlösungen waren die Parameterwerte: $ns = 14, nb = ns, ne = 4, nrb = 10, nre = 20$. Die Wahrscheinlichkeit, eine schlechtere Lösung bei der Suche in der Nachbarschaft zu akzeptieren, lag bei 10%. Allerdings wurde die bereits beste Lösung immer behalten (elitärer Algorithmus). Die maximale Iterationsanzahl betrug 10000, wobei der Stagnationswert zum Abbruch der Iteration ohne Verbesserung bei $\sqrt{anz}$ Schritten lag.

6.4 Auswertungen der Testreihen

Die Auswertungen der Testergebnisse, die sich anhand der in Abschnitt 6.3 vorgestellten Testreihen und Randbedingungen und hinsichtlich der in Abschnitt 6.1 aufgeführten Fragestellungen ergeben haben, werden zunächst für die deterministischen Verfahren in Abschnitt 6.4.1 vorgestellt. Dabei werden auch Empfehlungen zu besonders effizienten Verfahrensvariatnten gegeben. Anschließend werden in Abschnitt 6.4.2 die Ergebnisse für die bionischen Verfahren ausgewertet und eine vergleichende Gegenüberstellung vorgenommen.

6.4.1 Auswertungen zu den deterministischen Verfahren

Die Auswertungen der Testergebnisse beziehen sich zunächst auf die 35 Grundvarianten (18 *SucC*-Varianten, 16 *SimC*-Varianten (nur *impl*), einfaches Zickzack-Verfahren), jeweils ergänzt durch die 7 implementierten Methoden zur expliziten Hubbestimmung (vgl. Tabelle 11, Seite 100). Damit stehen in diesem ersten Untersuchungsschritt 245 Verfahrensvarianten im Vergleich, wobei das Augenmerk zunächst auf die primäre Zielgröße *Gesamtaufwand* in Bezug auf die entsprechenden Gütekriterien gerichtet ist (Abschnitt 6.4.1.1 - 6.4.1.3), während danach auch die sekundäre Zielgröße *Rechenaufwand* einbezogen wird und entsprechende Effizienzmessungen vorgenommen werden (Abschnitt 6.4.1.4). Zudem werden im Einzelnen drei verschiedene Einstellungen der Gewichtungsparameter betrachtet (siehe Seite 99).

Danach werden die beiden Zusatzmodule *Aussparung von Leerfeldern* (*REN_1*) und *Laubmengenobergrenzenvariation* (*OML_1*) betrachtet und Effizienzaussagen zu deren Einsatz getroffen (Abschnitt 6.4.1.5 - 6.4.1.6).

6.4.1.1 Häufigkeitsaussagen über beste Ergebnisse bzgl. Gesamtaufwand

In den folgenden Tabellen (Tabelle 12 - 14) sind die Verfahrensvarianten aufgelistet, welche zur jeweiligen Parametereinstellung am häufigsten das jeweils beste Ergebnis bei den 66 Testgärten erzielen.

Tabelle 12: *Häufigkeiten erster Platz (GA) bei Parametereinstellung (P1)*

ID	Verfahren	Häufigkeit TOP 1 (GA)
109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median	26
124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	8
107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_Median	7
119	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_Median	7
121	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median	5
112	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	3
115	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_Median	3
108	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_Median	1
110	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO_Median	1
111	SucC-LM_max-LIFO-LM_max-OML_0-REN_0-PO_Median	1
118	SucC-LM_min-LIFO-NW-OML_0-REN_0-PO_Median	1
120	SucC-NW-FIFO-LM_max-OML_0-REN_0-PO_Median	1
122	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_Median	1
126	SimC-NW-LM_max-Impl-OML_0-REN_0-PO_Median	1

Tabelle 13: *Häufigkeiten erster Platz (GA) bei Parametereinstellung (P2)*

ID	Verfahren	Häufigkeit TOP 1 (GA)
109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median	27
119	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_Median	9
121	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median	7
124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	7
107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_Median	6
115	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_Median	3
108	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_Median	2
110	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO_Median	2
112	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	2
122	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_Median	1

Tabelle 14: *Häufigkeiten erster Platz (GA) bei Parametereinstellung (P3)*

ID	Verfahren	Häufigkeit TOP 1 (GA)
109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median	20
121	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median	12
119	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_Median	9
107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_Median	8
124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	6
112	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	5
108	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_Median	2
110	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO_Median	2
115	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_Median	1
122	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_Median	1

Bei diesem Gütekriterium liegt eine deutliche Dominanz der *SucC*-Varianten vor, wobei sowohl die Breitensuche (*FIFO*) als auch die Tiefensuche (*LIFO*) anzutreffen ist. Bei der Repräsentantenwahl kommen vorrangig die *LMmax*-Methode und die *NW*-Methode vor, während bei der Auswahl des Aufnahmekandidaten die Methoden *NW* und *KPmin* in den oberen Positionen in Erscheinung treten. Darüber hinaus zeigt sich die eindeutige Dominanz der Median-Methode für die explizite Hubbestimmung. Insbesondere sticht die Variante *SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median* (109) mit häufigsten ersten Plätzen hervor. Weiterhin rangieren die folgenden Verfahrensvarianten stets ganz oben:

SucC-NW-FIFO-NW-OML_0-REN_0-PO_Median (121)
SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_Median (119)
SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median (124)
SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_Median (107)

Diese Varianten gehören dann erwartungsgemäß auch bzgl. des Gütekriteriums *Durchschnittliche Platzierung* zu den besten Verfahren, zeigen sich dabei allerdings unterschiedlich stabil, wenn man zusätzlich die Worst-Case-Platzierungen betrachtet (siehe Tabelle 15 - 17).

Insgesamt bestätigt sich auch hier die eindeutige Dominanz der *SucC*-Verfahren in Verbindung mit der Median-Methode für die explizite Hubbestimmung. Beispielsweise ist aus Tabelle 12 und Tabelle 15 unter anderem abzulesen, dass die Verfahrensvariante *SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median* (109) bei 26 der 66 Tests das beste Ergebnis (GA) erzielt, durchschnittlich Platz 2 und im schlechtesten Fall Platz 19 bei den insgesamt 245 betrachteten Varianten belegt (bei Parametereinstellung (P1)).

Tabelle 15: *Durchschnittliche und schlechteste Platzierung (GA) zur Parametereinstellung (P1); Durchschnittswerte $\leq$ 10*

ID	Verfahren	Durchschn. Platzierung	Worst-Case-Platzierung
109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median	2	19
121	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median	4	32
124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	4,5	47
107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_Median	5	34
112	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	5	46
119	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_Median	7	34
122	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_Median	8,5	37
110	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO_Median	9	32
115	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_Median	9	45

Tabelle 16: *Durchschnittliche und schlechteste Platzierung (GA) zur Parametereinstellung (P2); Durchschnittswerte $\leq$ 10*

ID	Verfahren	Durchschn. Platzierung	Worst-Case-Platzierung
109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median	2	32
121	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median	3	38
124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	4	52
112	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	4,5	46
107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_Median	6,5	43
119	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_Median	7	37
110	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO_Median	9,5	54

Tabelle 17: *Durchschnittliche und schlechteste Platzierung (GA) zur Parametereinstellung (P3); Durchschnittswerte $\leq$ 10*

ID	Verfahren	Durchschn. Platzierung	Worst-Case-Platzierung
109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median	2	35
121	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median	4	41
124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	4	52
112	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	5	54
107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_Median	7,5	49
119	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_Median	8,5	62

Bei Betrachtung der Platzierungshäufigkeiten unter den 5 besten Ergebniswerten (GA) ergibt sich ein ähnliches Bild (siehe Tabelle 18 - 20). Auch hier nehmen die Verfahrensvarianten mit ID 109 und 121 eine führende Stellung ein.

Tabelle 18: *Häufigkeiten der Platzierung unter den TOP 5 (GA) zur Parametereinstellung (P1); Häufigkeitswerte* ≥ 10

ID	Verfahren	Häufigkeit TOP 1 (GA)
109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median	49
121	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median	45
124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	37
107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_Median	36
112	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	34
119	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_Median	21
122	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_Median	19
115	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_Median	17
110	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO_Median	13
108	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_Median	11

Tabelle 19: *Häufigkeiten der Platzierung unter den TOP 5 (GA) zur Parametereinstellung (P2); Häufigkeitswerte* ≥ 10

ID	Verfahren	Häufigkeit TOP 1 (GA)
109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median	51
121	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median	45
124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	43
112	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	41
107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_Median	29
119	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_Median	21
115	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_Median	15
122	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_Median	12

Tabelle 20: *Häufigkeiten der Platzierung unter den TOP 5 (GA) zur Parametereinstellung (P3); Häufigkeitswerte* ≥ 10

ID	Verfahren	Häufigkeit TOP 1 (GA)
109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median	46
121	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median	46
112	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	40
124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	38
107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_Median	23
119	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_Median	21
115	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_Median	13
122	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_Median	10

6.4.1.2 Abweichungen von besten Ergebnissen bzgl. Gesamtaufwand

Eine weitere Bestätigung ihrer Lösungsgüte ergibt sich anhand des Gütekriteriums *Durchschnittliche prozentuale Abweichung* in Verbindung mit *Maximale prozentuale Abweichung* (siehe Tabelle 21 - 23). Beispielsweise zeigt sich in Tabelle 21, dass die Verfahrensvariante 109 eine durchschnittliche Abweichung um 1,05% von den besten Ergebnissen (GA) zu den 66 Testgärten aufweist. Im „schlimmsten“ Fall kommt es zu eine Abweichung um 7,86%. Demgegenüber weist die Variante 107 mit maximal 7,25% eine leicht bessere Abweichungsstabilität, schneidet aber im Schnitt (1,79%) schlechter als Variante 109 ab.

Tabelle 21: *Durchschnittliche (DPA) und maximale (MPA) prozentuale Abweichung vom besten Ergebnis (GA) bei Parametereinstellung (P1); Abweichungswerte* $\leq 5\%$

ID	Verfahren	DPA	MPA
109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median	1,05%	7,86%
121	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median	1,67%	10,37%
107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO _Median	1,79%	7,25%
124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	1,84%	12,10%
112	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	2,14%	12,48%
119	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_Median	2,16%	11,32%
122	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_Median	2,54%	8,41%
115	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_Median	2,83%	10,70%
110	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO _Median	3,02%	11,84%
118	SucC-LM_min-LIFO-NW-OML_0-REN_0-PO_Median	3,47%	15,98%
113	SucC-LM_min-FIFO-KP_min-OML_0-REN_0-PO _Median	3,75%	9,94%
116	SucC-LM_min-LIFO-KP_min-OML_0-REN_0-PO _Median	4,13%	10,48%

Tabelle 22: *Durchschnittliche (DPA) und maximale (MPA) prozentuale Abweichung vom besten Ergebnis (GA) bei Parametereinstellung (P2); Abweichungswerte* $\leq 5\%$

ID	Verfahren	DPA	MPA
109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median	1,28%	10,39%
121	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median	1,86%	11,93%
124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	2,24%	13,55%
112	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	2,41%	14,17%
107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO _Median	2,60%	14,48%
119	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_Median	2,90%	20,42%
122	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_Median	3,47%	11,20%
115	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_Median	3,56%	12,87%
110	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO _Median	3,93%	18,73%
118	SucC-LM_min-LIFO-NW-OML_0-REN_0-PO_Median	4,42%	18,36%
113	SucC-LM_min-FIFO-KP_min-OML_0-REN_0-PO _Median	4,89%	13,99%
39	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_MaxLaub	4,89%	16,37%

Tabelle 23: *Durchschnittliche (DPA) und maximale (MPA) prozentuale Abweichung vom besten Ergebnis (GA) bei Parametereinstellung (P3); Abweichungswerte* $\leq 5\%$

ID	Verfahren	DPA	MPA
109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median	1,46%	11,29%
121	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median	2,00%	12,45%
124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	2,52%	14,06%
112	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	2,61%	14,71%
107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO _Median	3,12%	20,09%
119	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_Median	3,39%	27,76%
115	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_Median	3,99%	13,70%
122	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_Median	3,99%	14,37%
39	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_MaxLaub	4,32%	16,44%
110	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO _Median	4,48%	25,01%
51	SucC-NW-FIFO-NW-OML_0-REN_0-PO_MaxLaub	4,92%	13,55%
118	SucC-LM_min-LIFO-NW-OML_0-REN_0-PO_Median	4,93%	19,15%

6.4.1.3 Überdeckungsmengen bzgl. Gesamtaufwand

Da zu den 66 Testgärten verschiedene Verfahren das jeweils beste Ergebnis (niedrigster Gesamtaufwand) liefern, stellt sich die Frage, wie viele und welche von den insgesamt 245 Verfahrensvarianten (ohne Einbeziehung der Zusatzmodule, also *OML_0-REN_0*) dazu ausreichen, sodass aus ihrer Mitte stets die beste Lösung dabei ist. Bei allen drei Parametereinstellungen ist die jeweilige komplette Überdeckungsmenge identisch mit der entsprechenden Liste der Verfahrensvarianten, die eine positive Häufigkeit des 1. Platzes aufweisen (siehe Tabelle 12 - 14). Aufgrund dieses Sachverhaltes sind diese Überdeckungsmengen sogar eindeutig. Da zudem gewährleistet ist, dass die angegebenen Auswahlen irreduzibel sind, d.h. bei Streichung einer Verfahrensvariante liegt keine komplette Überdeckung des 1. Platzes bzgl. Gesamtaufwand für alle Testgärten mehr vor. Da in diesem Falle auch kein Verfahrensaustausch zu einer anderen kompletten Überdeckungsmenge führen kann, zeigt sich beispielsweise, dass durchaus auch *SimC*-Varianten eine wichtige „Lückenbüßerrolle“ spielen können (hierzu siehe die Variante 126 in Tabelle 12).

Es sei an dieser Stelle aber darauf hingewiesen, dass die hier beobachtete Übereinstimmung von „Liste 1. Platz“ mit der kompletten Überdeckungsmenge nicht immer gegeben ist, vielmehr kann es vorkommen, dass zur Bildung einer kompletten Überdeckungsmenge auf eine Vielzahl von Varianten mit positiven Häufigkeitswerten verzichtet werden kann. Diesbezüglich sei hier bereits auf die spätere Einbeziehung der Zusatzmodule verwiesen, wenn 490 bzw. 980 verschiedene Varianten verglichen werden (vgl. Abschnitt 6.4.1.5 und 6.4.1.6).

6.4.1.4 Häufigkeitsaussagen über beste Effizienzwerte

Im Gegensatz zu den bisherige Auswertungen allein anhand der primären Ergebnisgröße *Gesamtaufwand* werden nun durch die Einbeziehung der sekundären Ergebnisgröße *Rechenaufwand* Effizienzuntersuchungen angestellt.

Um sich eine grobe Einschätzung des Rechenaufwandes machen zu können, sollen hier die Rechenzeiten für zwei ausgewählte Grundtypen angegeben werden. Das einfache Zickzack-Verfahren [1] benötigt selbst für große Gärten (100×100) weniger als 0,05 Sekunden. Das *SucC*-Verfahren [4] benötigt in Abhängigkeit der Gartengröße ca. 0,15 sec. für kleine (25×25), ca. 2 sec. für mittlere (50×50) bzw. 12 sec. für große Gärten. Allerdings erhöht die Hinzunahme einer expliziten Hubbestimmung, z.B. der Median-Methode, den Rechenaufwand beträchtlich. Für die Zickzack-Variante [106] erhöht sich die Rechenzeit auf bis zu 2 sec. für kleine, auf bis zu 12 sec. für mittlere und auf bis zu 43 sec. für große Gärten. In ähnlicher Weise gilt dieser Zuwachs auch für die *SucC*-Variante [109]: bis 2 sec. für kleine, bis 20 sec. für mittlere, bis 52 sec. für große Gärten.

Anhand dieser Größenordnungen wird ersichtlich, dass der Rechenaufwand in praktischen Fällen eine durchaus mitentscheidende Rolle spielen kann, wozu Effizienzwerte als Entscheidungshilfe dienen können.

Die Effizienzwerte der einzelnen Verfahren bei den jeweiligen Testgärten erfolgt in Anlehnung an den Vorschlag im Anhang A.1. Es werden die Rechenzeit des schnellsten Verfahrens V^* und der zugehörige Gesamtaufwand als Referenzwerte genommen: $RZ(V^*)$ bzw. $GA(V^*)$. Die „Mehrzeit“ eines Verfahrens V (d.h. $RZ(V) - RZ(V^*)$) wird ins Verhältnis zur Verbesserung des Gesamtaufwandes ($GA(V^*) - GA(V)$) gesetzt, wobei sich die „Wichtigkeit“ des Rechenaufwandes anhand des Effizienzparameters α steuern lässt ($\alpha \geq 0$):

$$EW(V) = \frac{GA(V^*) - GA(V)}{[RZ(V) - RZ(V^*)]^\alpha}. \tag{6.1}$$

Diese Effizienzwerte lassen sich noch dahingehend normieren, dass beispielsweise der größte Effizienzwert bzgl. des jeweiligen Testgartens auf 100% gesetzt wird. Das Ranking der Verfahren hinsichtlich ihrer Effizienzwerte erfolgt analog zum Ranking der Häufigkeiten für beste Lösungen (TOP 1 oder TOP x).

Tabelle 24: *Häufigkeiten der besten Effizienzen (TOP 1) zu verschiedenen Werten für den Effizienzparameter α bei Aufwandsparametereinstellung (P1)*

	Häufigkeit beste Effizienz zum Effizienzparameter α										
ID	**0,00**	**0,05**	**0,10**	**0,20**	**0,30**	**0,50**	**1,00**	**1,50**	**2,00**	**3,00**	**5,00**
16	0	0	4	9	11	13	15	13	12	12	9
19	0	0	0	1	1	4	4	8	9	9	12
85	0	2	5	6	5	5	3	2	2	2	2
107	7	6	1	1	0	0	0	0	0	0	0
109	26	25	13	6	6	5	3	2	2	2	2
119	7	5	1	1	2	2	3	2	2	2	3
121	5	5	2	1	0	1	0	0	0	0	0
124	8	7	5	2	2	2	2	3	2	2	1
143	0	1	1	1	3	2	3	4	4	4	4
155	0	2	7	7	7	6	3	2	2	2	2
158	0	0	1	3	3	4	5	4	4	4	4
189	0	1	4	3	3	1	1	1	1	1	1
190	0	2	7	8	7	8	6	5	5	5	5

In Tabelle 24 wird das Verhalten von einigen Verfahren mit hohen Häufigkeitswerten in Abhängigkeit vom Effizienzparameter α dargestellt (exemplarisch zur Parametereinstellung (P1)). Die grün unterlegten Felder in der Tabelle zeigen die jeweilige maximale Effizienzhäufigkeit an. Die Einstellung $\alpha = 0$ entspricht den Häufigkeitswerten ohne Einbeziehung des Rechenaufwandes (vgl. Abschnitt 6.4.1.1). Erwartungsgemäß zeigt sich, dass einige Verfahren mit steigenden Effizienzparameterwerten an Bedeutung verlieren, andere an Bedeutung gewinnen, wobei die Bedeutung hier anhand der Häufigkeiten bester Effizienzwerte gemessen wird.

Mit wachsender Einflussnahme der sekundären Ergebnisgröße ($\alpha > 0$) tritt anfänglich eine Verbreiterung der besten Effizienzwerte auf mehr Verfahren ein. Haben im Falle $\alpha = 0$ nur 14 von 245 Verfahren positive Häufigkeitswerte (d.h. dass sie zu mindestens einem der 66 Testgärten den besten Effizienzwert aufweisen), so ist deren Anzahl bei $\alpha = 0,05$ auf 24 und bei $\alpha = 0,1$ auf 47 angestiegen. Für die übrigen untersuchten Parameterwerte (von $\alpha = 0,2$ bis $\alpha = 5$) bleibt diese Anzahl nahezu konstant um 40. Erst bei sehr hoher Gewichtung nimmt die Anzahl wieder ab und im Extremfall, dass einem Entscheidungsträger allein an einer möglichst kurzen Rechenzeit gelegen ist, verdichtet sich die beste Effizienz auf das einfache (weil schnellste) Zickzack-Verfahren (*Zickzack-OML_0-REN_0-PO_None*).

Im Verlauf der zunehmenden Gewichtung des Rechenaufwandes treten auch andere Verfahren in Erscheinung, die zuvor keine Rolle gespielt haben, sondern erst mit wachsender Gewichtung des Rechenaufwandes an Bedeutung gewin-

nen. Um festzustellen, welche Verfahrensvarianten ohne eine Festlegung der Gewichtung der beiden Ergebnisgrößen *Gesamtaufwand* und *Rechenaufwand* überhaupt beste Effizienzwerte erzielen können, bietet sich die Ermittlung der Pareto-Auswahlmenge an.[81]

In Tabelle 25 sind die 83 Verfahrensvarianten aufgeführt, die mindestens einmal zu einer Pareto-Auswahlmenge der 66 Testgärten gehören (exemplarisch zur Parametereinstellung (P1)). Daraus ist auch die nicht minder informative Aussage abzuleiten, dass den übrigen 162 der 245 Verfahrensvarianten keine Bedeutung im Hinblick auf Effizienz zugewiesen werden kann, da sie gewichtungsunabhängig (d.h. für beliebige α-Werte) keine beste Effizienz bei den Untersuchungen erreicht haben und somit auch für andere ähnliche Gärten nicht erreichen dürften.

Unter den 83 aufgeführten Varianten ist der Grundtyp *SucC* vorrangig vertreten, vereinzelt kommen aber auch *Zickzack*- und *SimC*-Varianten vor. Darüber hinaus ergibt sich ein vollständiges Mix aller 7 Methoden zur expliziten Hubbestimmung, wodurch ihre spezielle Einsatzberechtigung legitimiert wird.

6.4.1.5 Zur Effizienz der Aussparung von laubleeren Feldern

Das Zusatzmodul *Aussparung von laubleeren Feldern* kann immer dann zu Einsparungen führen, wenn bei der Laubmengenverteilung im Garten viele Felder leer bleiben ($M(a) = 0$). Da die Basisversionen der deterministischen Heuristiken auch laubleere Felder grundsätzlich als beharkbar ansehen, können erzeugte Harkvorschriften auch leere Harkquellen[82] besitzen. Zwar entsteht durch das Harken einer „Nullmenge“ kein Hark-, wohl aber ein Wegeaufwand, der eliminiert werden könnte. Mit dem Zusatzmodul werden im Nachgang in jedem Cluster leere Felder ermittelt und in einelementige Cluster umgewandelt, wodurch sie der Aufwandsberechnung entzogen werden.[83]

Das Einsparungspotential im Hinblick auf die Gesamtkosten wird in Tabelle 26 ersichtlich. Hier sind sinnvollerweise nur die Testgärten mit leeren Feldern aufgeführt. Vereinzelt ergeben sich Einsparungen über 4% des Gesamtaufwandes.

[81] Hierzu siehe Seite 96 und Anhang A.1.
[82] Zum Begriff der Harkquelle siehe Seite 18.
[83] Näheres siehe in Abschnitt 6.3, Seite 101.

Tabelle 25: *Häufigkeit der Zugehörigkeit zur Pareto-Auswahlmenge bei Parametereinstellung (P1)*

ID	Verfahren	Häufigkeit der Zugehörigkeit zur Paretomenge	ID	Verfahren	Häufigkeit der Zugehörigkeit zur Paretomenge
1	Zickzack-OML_0-REN_0-PO_None	41	177	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_SmallestIndex	4
109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median	32	17	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_None	4
121	SucC-NW-FIFO-NW-OML_0-REN_0-PO_Median	25	120	SucC-NW-FIFO-LM_max-OML_0-REN_0-PO_Median	3
39	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_MaxLaub	20	108	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_Median	3
124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	19	53	SucC-NW-LIFO-LM_max-OML_0-REN_0-PO_MaxLaub	3
176	Zickzack-OML_0-REN_0-PO_SmallestIndex	19	88	SucC-NW-LIFO-LM_max-OML_0-REN_0-PO_MinLaub	3
16	SucC-NW-FIFO-NW-OML_0-REN_0-PO_None	18	156	SucC-NW-FIFO-NW-OML_0-REN_0-PO_MinKompost	3
54	SucC-NW-LIFO-NW-OML_0-REN_0-PO_MaxLaub	17	159	SucC-NW-LIFO-NW-OML_0-REN_0-PO_MinKompost	3
190	SucC-NW-FIFO-LM_max-OML_0-REN_0-PO_SmallestIndex	16	192	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_SmallestIndex	3
51	SucC-NW-FIFO-NW-OML_0-REN_0-PO_MaxLaub	15	180	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO_SmallestIndex	3
19	SucC-NW-LIFO-NW-OML_0-REN_0-PO_None	14	110	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO_Median	2
112	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median	13	116	SucC-LM_min-LIFO-KP_min-OML_0-REN_0-PO_Median	2
119	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_Median	13	40	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO_MaxLaub	2
155	SucC-NW-FIFO-LM_max-OML_0-REN_0-PO_MinKompost	13	84	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_MinLaub	2
15	SucC-NW-FIFO-LM_max-OML_0-REN_0-PO_None	11	79	SucC-LM_min-FIFO-LM_max-OML_0-REN_0-PO_MinLaub	2
85	SucC-NW-FIFO-LM_max-OML_0-REN_0-PO_MinLaub	11	76	SucC-LM_max-LIFO-LM_max-OML_0-REN_0-PO_MinLaub	2
122	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_Median	10	181	SucC-LM_max-LIFO-LM_max-OML_0-REN_0-PO_SmallestIndex	2
178	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_SmallestIndex	10	72	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_MinLaub	2
18	SucC-NW-LIFO-LM_max-OML_0-REN_0-PO_None	10	87	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_MinLaub	2
52	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_MaxLaub	9	113	SucC-LM_min-FIFO-KP_min-OML_0-REN_0-PO_Median	1
225	SucC-NW-FIFO-LM_max-OML_0-REN_0-PO_KeepHubs	9	123	SucC-NW-LIFO-LM_max-OML_0-REN_0-PO_Median	1
107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_Median	8	111	SucC-LM_max-LIFO-LM_max-OML_0-REN_0-PO_Median	1
50	SucC-NW-FIFO-LM_max-OML_0-REN_0-PO_MaxLaub	8	126	SimC-NW-LM_max-Impl-OML_0-REN_0-PO_Median	1
49	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_MaxLaub	7	43	SucC-LM_min-FIFO-KP_min-OML_0-REN_0-PO_MaxLaub	1
3	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_None	7	45	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_MaxLaub	1
73	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_MinLaub	7	48	SucC-LM_min-LIFO-NW-OML_0-REN_0-PO_MaxLaub	1
191	SucC-NW-FIFO-NW-OML_0-REN_0-PO_SmallestIndex	7	41	SucC-LM_max-LIFO-LM_max-OML_0-REN_0-PO_MaxLaub	1
42	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_MaxLaub	6	216	SucC-LM_max-LIFO-LM_max-OML_0-REN_0-PO_KeepHubs	1
38	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_MaxLaub	6	9	SucC-LM_min-FIFO-LM_max-OML_0-REN_0-PO_None	1
143	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_MinKompost	6	149	SucC-LM_min-FIFO-LM_max-OML_0-REN_0-PO_MinKompost	1
228	SucC-NW-LIFO-LM_max-OML_0-REN_0-PO_KeepHubs	6	226	SucC-NW-FIFO-NW-OML_0-REN_0-PO_KeepHubs	1
158	SucC-NW-LIFO-LM_max-OML_0-REN_0-PO_MinKompost	6	47	SucC-LM_min-LIFO-LM_max-OML_0-REN_0-PO_MaxLaub	1
189	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_SmallestIndex	6	2	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_None	1
115	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_Median	5	12	SucC-LM_min-LIFO-LM_max-OML_0-REN_0-PO_None	1
37	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_MaxLaub	5	82	SucC-LM_min-LIFO-LM_max-OML_0-REN_0-PO_MinLaub	1
14	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_None	5	187	SucC-LM_min-LIFO-LM_max-OML_0-REN_0-PO_SmallestIndex	1
118	SucC-LM_min-LIFO-NW-OML_0-REN_0-PO_Median	4	150	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_MinKompost	1
213	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_KeepHubs	4	194	SucC-NW-LIFO-NW-OML_0-REN_0-PO_SmallestIndex	1
193	SucC-NW-LIFO-LM_max-OML_0-REN_0-PO_SmallestIndex	4	5	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO_None	1
184	SucC-LM_min-FIFO-LM_max-OML_0-REN_0-PO_SmallestIndex	4	134	SimC-LM_min-LM_max-Impl-OML_0-REN_0-PO_Median	1
6	SucC-LM_max-LIFO-LM_max-OML_0-REN_0-PO_None	4	71	Zickzack-OML_0-REN_0-PO_MinLaub	1
146	SucC-LM_max-LIFO-LM_max-OML_0-REN_0-PO_MinKompost	4			

Tabelle 26: *Einsparung durch Einsatz des Zusatzmoduls Aussparung laubleerer Felder bei Parametereinstellung (P1); geordnet nach prozentualer Einsparung*

TestCaseID	Beste Lösung OHNE Zusatzmodul	Beste Lösung MIT Zusatzmodul	Einsparung absolut	Einsparung prozentual
28	12318,83	11815,99	502,84	4,082%
10	2346,37	2251,35	95,01	4,049%
19	5290,77	5089,72	201,05	3,800%
1	1040,66	1007,46	33,20	3,190%
37	30396,04	29684,21	711,83	2,342%
154	39741,12	39157,19	583,93	1,469%
136	14440,94	14240,24	200,69	1,390%
145	31930,70	31585,38	345,32	1,081%
127	13453,42	13336,51	116,91	0,869%
46	106532,48	105670,72	861,76	0,809%
55	317372,29	315618,15	1754,14	0,553%
190	80557,50	80291,24	266,27	0,331%
73	3453,62	3442,79	10,83	0,314%
91	19224,65	19166,33	58,33	0,303%
82	7806,75	7784,37	22,38	0,287%
172	22794,93	22762,55	32,38	0,142%
163	19884,17	19871,68	12,49	0,063%
100	48232,42	48220,02	12,40	0,026%
118	591442,79	591297,11	145,68	0,025%
181	55726,21	55726,17	0,04	0,000%
109	191174,67	191174,67	0,00	0,000%

Da sich die Einsparungen ausschließlich auf die Wegekosten beziehen, ist das Einsparungspotential umso geringer, je niedriger der Wegeaufwand gewichtet ist (α_W). Bei den Parametereinstellungen (P2) und (P3) ergeben sich nur noch Einsparungen unter 1,75% bzw. 0,71%.

Da die Rechenzeit für das Zusatzmodul verschwindend gering ausfällt, ist sein Einsatz eine durchaus effiziente Verbesserungsmöglichkeit. Diese Befürwortung wird auch nicht entscheidend durch den Einwand geschmälert, dass es sehr vereinzelte Fälle gibt, bei denen sich die Lösung durch den Einsatz dieses Zusatzmoduls verschlechtert. Daher ist man komplett „auf der sicheren Seite“, wenn man jede ausgewählte Verfahrensvariante sowohl mit als auch ohne Zusatzmodul aufruft (*REN_1* und *REN_0*).

Die Effizienz des Zusatzmoduls zeigt sich auch in der Bildung der kompletten Überdeckungsmenge, die in Tabelle 27 exemplarisch zur Parametereinstellung (P1) präsentiert wird (in aufsteigender ID-Reihenfolge).[84]

Tabelle 27: *Durchschnittliche Platzierung (DPI) und durchschnittliche prozentuale Abweichung (DPA) der komplette Überdeckungsmenge bei Einsatz des Zusatzmoduls Aussparung laubleerer Felder zur Parametereinstellung (P1); aufsteigend geordnet nach ID*

ID	Verfahren	Häufigkeit TOP 1	DPI	DPA
112	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median	3	6	2,54%
352	SucC-LM_max-FIFO-KP_min-OML_0-REN_1-PO_Median	7	5	1,80%
353	SucC-LM_max-FIFO-LM_max-OML_0-REN_1-PO_Median	1	22,5	5,94%
354	SucC-LM_max-FIFO-NW-OML_0-REN_1-PO_Median	26	2	1,05%
355	SucC-LM_max-LIFO-KP_min-OML_0-REN_1-PO_Median	1	11	3,11%
356	SucC-LM_max-LIFO-LM_max-OML_0-REN_1-PO_Median	1	26,5	7,02%
360	SucC-LM_min-FIFO-NW-OML_0-REN_1-PO_Median	3	10	2,82%
363	SucC-LM_min-LIFO-NW-OML_0-REN_1-PO_Median	1	13	3,49%
364	SucC-NW-FIFO-KP_min-OML_0-REN_1-PO_Median	7	8	2,12%
365	SucC-NW-FIFO-LM_max-OML_0-REN_1-PO_Median	1	18	5,49%
366	SucC-NW-FIFO-NW-OML_0-REN_1-PO_Median	5	4,5	1,69%
367	SucC-NW-LIFO-KP_min-OML_0-REN_1-PO_Median	1	9	2,56%
369	SucC-NW-LIFO-NW-OML_0-REN_1-PO_Median	7	4	1,84%
371	SimC-NW-LM_max-Impl-OML_0-REN_1-PO_Median	1	48	7,67%
379	SimC-LM_min-LM_max-Impl-OML_0-REN_1-PO_Median	1	84,5	14,65%

Hierin sind der Großteil der Verfahrensvarianten mit dem Zusatzmodul *REN_1* versehen (mit Ausnahme ID 112). Die komplette Überdeckungsmenge, die sich gänzlich ohne Zusatzmodul (*REN_0*) ergibt (vgl. Tabelle 12), überdeckt nur 46 der 66 Testgärten (knapp 70%), gemessen an den verbesserten Lösungen aufgrund des Einsatzes des Zusatzmoduls, wobei eine durchschnittliche prozentuale Abweichung von den besten Ergebnissen um 0,39% entsteht. Betrachtet man aber speziell diejenigen Testgärten, die aufgrund laubleerer Felder für den Einsatz des Zusatzmoduls prädestiniert sind, so werden davon durch Verfahren ohne Zusatzmodul (*REN_0*) nur noch 9,1% mit besten Ergebnissen überdeckt (vgl. Tabelle 26).

6.4.1.6 Zur Effizienz der Laubmengenobergrenzenvariation

Die Untersuchungen an der Güte des Zusatzmoduls *Laubmengenobergrenzenvariation*, basierend auf den in Abschnitt 6.3 festgelegten Grundlagen, lässt keine grundsätzliche Befürwortung oder Ablehnung von dessen Einsatz zu. Zunächst sollen hier die „Plus- und Minuspunkte“ aufgeführt werden, bevor vorsichtige

[84] Die kompletten Überdeckungsmengen zu den Parametereinstellungen (P2) und (P3) findet man in Tabelle 33, Seite 123.

Empfehlungen ausgesprochen werden.

Für den Einsatz spricht zunächst der Sachverhalt, dass sich bei über 60% aller Fälle[85] Verbesserungen beim Vergleich der Ergebnisse (GA) zeigen (ohne bzw. mit Laubmengenobergrenzenvariation). Von besonderem Interesse aber sind die Vergleiche zu den jeweils besten Lösungen. Hier zeigen sich Verbesserungen bei 11 bzw. 2 bzw. 1 der 66 Testgärten zu den Parametereinstellungen (P1) - (P3), entspricht 16,67% bzw. 3,03% bzw. 1, 51%. Diese Verbesserungen fallen meistens sehr gering aus, im Schnitt unter 1%, im Einzelfall aber über 5%.

Diesen Einsparungen an den Gesamtkosten stehen die Erhöhungen des Rechenaufwandes gegenüber, die sich an der Anzahl der Aufrufe abschätzen lassen. Wenn beispielsweise die maximale Laubmengenobergrenze $\overline{M}$ fiktiv in 10 Schritten herabgesetzt wird, ergibt sich ungefähr der zehnfache Rechenaufwand pro Garten und Verfahren.

Somit stellt sich die Frage nach der Effizienz für den Einsatz dieses Zusatzmoduls. Hier ist der Entscheidungsträger gefragt, ob der erhöhte Rechenaufwand die potentielle Verringerung des Gesamtaufwandes rechtfertigt.

Der Vollständigkeit halber werden in diesem Zusammenhang die jeweils besten Lösungen, die sich beim Einsatz der beiden Zusatzmodule *Aussparung laubleerer Felder* und *Laubmengenobergrenzenvariation* ergeben, in den folgenden Tabellen aufgelistet (Tabelle 29 - 31). Diese Ergebnisse können mit den entsprechenden Tabellen ohne deren Einsatz aus den Abschnitten 6.4.1.1 - 6.4.1.3 verglichen werden, wobei hier exemplarisch nur die Ergebnisse zur Parametereinstellung (P1) vorgestellt werden.

Auf den ersten Blick fällt die breitere Streuung der besten Ergebnisse über mehr Verfahrensvarianten auf (vgl. Tabelle 12). Dies erklärt sich dadurch, dass mehr Verfahrenvarianten (980 statt 245) um Platz 1 konkurrieren. Zudem ist festzuhalten, dass mehrere gleich gute Ergebnisse den jeweiligen 1. Platz einnehmen. So belegt beispielsweise das Grundverfahren *SucC-LM_max-FIFO-KP_min-OML_∗-REN_∗-PO_Median* in der Häufigkeitstabelle die ersten vier Stellen, wobei in nachvollziehbarer Weise die Untervarianten *OML_1* vor *OML_0* und *REN_1* vor *REN_0* rangieren. Darüber hinaus kommen 5 weitere Grundverfahren aus Tabelle 12 in allen 4 *OML/REN*-Versionen auch in der erweiterten Häufigkeitstabelle vor, während die übrigen *SucC*-Grundverfahren aus Tabelle 12 zumindest in der

[85] Hiermit sind 194040 Einzelergebnisse (GA) gemeint, die sich aus 980 Verfahrensvarianten und 66 Testgärten bei 3 verschiedenen Parametereinstellungen ergeben.

OML_1/REN_1-Form in der erweiterten Häufigkeitstabelle vertreten sind.

Dieser Sachverhalt, dass nahezu alle Grundverfahren aus der Häufigkeitstabelle Tabelle 12 auch in der erweiterten Häufigkeitstabelle (Tabelle 29) vertreten sind, zeugt von einer starken Gütestabilität dieser Grundverfahren. Zudem werden die Aussagen zur Dominanz der *SucC*-Varianten und der Median-Methode voll bestätigt (vgl. Seite 104).

Beim Vergleich der Häufigkeitstabellen (Tabelle 15 und Tabelle 28) zeigt sich eine Überstimmung in der Besetzung des oberen Ranges durch das Grundverfahren *SucC-LM_max-FIFO-NW-OML_∗-REN_∗-PO_Median* [109 bzw. 844].

Tabelle 28: *Durchschnittliche und schlechteste Platzierung (GA) beim Vergleich aller 980 Varianten zur Parametereinstellung (P1); Durchschnittswerte* ≤ 10

ID	Verfahren	Durchschnittl. Platzierung	Worst-Case-Platzierung
844	SucC-LM_max-FIFO-NW-OML_1-REN_1-PO_Median	2	24
354	SucC-LM_max-FIFO-NW-OML_0-REN_1-PO_Median	2,5	62
599	SucC-LM_max-FIFO-NW-OML_1-REN_0-PO_Median	3	57
856	SucC-NW-FIFO-NW-OML_1-REN_1-PO_Median	4,5	55
859	SucC-NW-LIFO-NW-OML_1-REN_1-PO_Median	4,5	61
369	SucC-NW-LIFO-NW-OML_0-REN_1-PO_Median	5	61
366	SucC-NW-FIFO-NW-OML_0-REN_1-PO_Median	5	76
614	SucC-NW-LIFO-NW-OML_1-REN_0-PO_Median	5,5	74
842	SucC-LM_max-FIFO-KP_min-OML_1-REN_1-PO_Median	6	45
611	SucC-NW-FIFO-NW-OML_1-REN_0-PO_Median	6	56
847	SucC-LM_max-LIFO-NW-OML_1-REN_1-PO_Median	6	69
352	SucC-LM_max-FIFO-KP_min-OML_0-REN_1-PO_Median	6	76
109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median	6	124
121	SucC-NW-FIFO-NW-OML_0-REN_0-PO_Median	6	149
602	SucC-LM_max-LIFO-NW-OML_1-REN_0-PO_Median	6,5	88
357	SucC-LM_max-LIFO-NW-OML_0-REN_1-PO_Median	6,5	115
597	SucC-LM_max-FIFO-KP_min-OML_1-REN_0-PO_Median	7	91
107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_Median	7	120
112	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median	7	154
854	SucC-NW-FIFO-KP_min-OML_1-REN_1-PO_Median	8	38
124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	8	74
364	SucC-NW-FIFO-KP_min-OML_0-REN_1-PO_Median	8	76
609	SucC-NW-FIFO-KP_min-OML_1-REN_0-PO_Median	9	97
119	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_Median	9	170
857	SucC-NW-LIFO-KP_min-OML_1-REN_1-PO_Median	9,5	72
850	SucC-LM_min-FIFO-NW-OML_1-REN_1-PO_Median	10	89

Tabelle 29: *Häufigkeiten 1. Platz (GA) beim Vergleich aller 980 Varianten bei Parametereinstellung (P1)*

ID	Verfahren	Häufigkeit TOP 1 (GA)
844	SucC-LM_max-FIFO-NW-OML_1-REN_1-PO_Median	28
354	SucC-LM_max-FIFO-NW-OML_0-REN_1-PO_Median	23
599	SucC-LM_max-FIFO-NW-OML_1-REN_0-PO_Median	23
109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median	18
364	SucC-NW-FIFO-KP_min-OML_0-REN_1-PO_Median	7
842	SucC-LM_max-FIFO-KP_min-OML_1-REN_1-PO_Median	7
854	SucC-NW-FIFO-KP_min-OML_1-REN_1-PO_Median	7
352	SucC-LM_max-FIFO-KP_min-OML_0-REN_1-PO_Median	6
369	SucC-NW-LIFO-NW-OML_0-REN_1-PO_Median	6
597	SucC-LM_max-FIFO-KP_min-OML_1-REN_0-PO_Median	6
856	SucC-NW-FIFO-NW-OML_1-REN_1-PO_Median	6
859	SucC-NW-LIFO-NW-OML_1-REN_1-PO_Median	6
107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_Median	5
124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	5
614	SucC-NW-LIFO-NW-OML_1-REN_0-PO_Median	5
119	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_Median	4
366	SucC-NW-FIFO-NW-OML_0-REN_1-PO_Median	4
609	SucC-NW-FIFO-KP_min-OML_1-REN_0-PO_Median	4
602	SucC-LM_max-LIFO-NW-OML_1-REN_0-PO_Median	3
112	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median	2
360	SucC-LM_min-FIFO-NW-OML_0-REN_1-PO_Median	2
610	SucC-NW-FIFO-LM_max-OML_1-REN_0-PO_Median	2
847	SucC-LM_max-LIFO-NW-OML_1-REN_1-PO_Median	2
850	SucC-LM_min-FIFO-NW-OML_1-REN_1-PO_Median	2
855	SucC-NW-FIFO-LM_max-OML_1-REN_1-PO_Median	2
115	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_Median	1
120	SucC-NW-FIFO-LM_max-OML_0-REN_0-PO_Median	1
353	SucC-LM_max-FIFO-LM_max-OML_0-REN_1-PO_Median	1
355	SucC-LM_max-LIFO-KP_min-OML_0-REN_1-PO_Median	1
356	SucC-LM_max-LIFO-LM_max-OML_0-REN_1-PO_Median	1
357	SucC-LM_max-LIFO-NW-OML_0-REN_1-PO_Median	1
365	SucC-NW-FIFO-LM_max-OML_0-REN_1-PO_Median	1
367	SucC-NW-LIFO-KP_min-OML_0-REN_1-PO_Median	1
605	SucC-LM_min-FIFO-NW-OML_1-REN_0-PO_Median	1
611	SucC-NW-FIFO-NW-OML_1-REN_0-PO_Median	1
613	SucC-NW-LIFO-LM_max-OML_1-REN_0-PO_Median	1
616	SimC-NW-LM_max-Impl-OML_1-REN_0-PO_Median	1
843	SucC-LM_max-FIFO-LM_max-OML_1-REN_1-PO_Median	1
845	SucC-LM_max-LIFO-KP_min-OML_1-REN_1-PO_Median	1
846	SucC-LM_max-LIFO-LM_max-OML_1-REN_1-PO_Median	1
857	SucC-NW-LIFO-KP_min-OML_1-REN_1-PO_Median	1
858	SucC-NW-LIFO-LM_max-OML_1-REN_1-PO_Median	1
861	SimC-NW-LM_max-Impl-OML_1-REN_1-PO_Median	1

Eine Bestätigung der angesprochenen Gütestabilität zeigt sich beim Vergleich der TOP-5-Häufigkeitstabellen in Tabelle 18 und Tabelle 30. Dort finden sich alle oben rangierten Grundverfahren in beiden Häufigkeitstabellen wieder, wenn auch mit leichten Reihenfolgeverschiebungen, und zwar:

- *SucC-LM_max-FIFO-NW-OML_∗-REN_∗-PO_Median* [109, 354, 599, 844],
- *SucC-NW-FIFO-NW-OML_∗-REN_∗-PO_Median* [121, 366, 611, 856],
- *SucC-NW-LIFO-NW-OML_∗-REN_∗-PO_Median* [124, 369, 614, 859],
- *SucC-LM_max-FIFO-KP_min-OML_∗-REN_∗-PO_Median* [107, 352, 589, 842],
- *SucC-NW-FIFO-NW-OML_∗-REN_∗-PO_Median* [112, 357, 602, 847].

Tabelle 30: *Häufigkeiten der Platzierung unter den TOP 5 (GA) beim Vergleich aller 980 Varianten zur Parametereinstellung (P1); Häufigkeitswerte* ≥ 20

ID	Verfahren	Häufigkeit TOP 5 (GA)
844	SucC-LM_max-FIFO-NW-OML_1-REN_1-PO_Median	50
354	SucC-LM_max-FIFO-NW-OML_0-REN_1-PO_Median	43
856	SucC-NW-FIFO-NW-OML_1-REN_1-PO_Median	40
859	SucC-NW-LIFO-NW-OML_1-REN_1-PO_Median	39
599	SucC-LM_max-FIFO-NW-OML_1-REN_0-PO_Median	38
366	SucC-NW-FIFO-NW-OML_0-REN_1-PO_Median	36
369	SucC-NW-LIFO-NW-OML_0-REN_1-PO_Median	34
614	SucC-NW-LIFO-NW-OML_1-REN_0-PO_Median	33
611	SucC-NW-FIFO-NW-OML_1-REN_0-PO_Median	32
842	SucC-LM_max-FIFO-KP_min-OML_1-REN_1-PO_Median	32
109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median	31
352	SucC-LM_max-FIFO-KP_min-OML_0-REN_1-PO_Median	31
121	SucC-NW-FIFO-NW-OML_0-REN_0-PO_Median	29
847	SucC-LM_max-LIFO-NW-OML_1-REN_1-PO_Median	29
124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	28
597	SucC-LM_max-FIFO-KP_min-OML_1-REN_0-PO_Median	26
602	SucC-LM_max-LIFO-NW-OML_1-REN_0-PO_Median	26
107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_Median	25
357	SucC-LM_max-LIFO-NW-OML_0-REN_1-PO_Median	25
112	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median	22
364	SucC-NW-FIFO-KP_min-OML_0-REN_1-PO_Median	21
854	SucC-NW-FIFO-KP_min-OML_1-REN_1-PO_Median	21

Letztendlich kommt noch der Vergleich der kompletten Überdeckungsmenge in Tabelle 12 und Tabelle 31 hinzu.[86] Beide Überdeckungsmengen stimmen komplett

[86] Wie in Abschnitt 6.4.1.3 angesprochen, besteht die komplette Überdeckungsmenge zum Untersuchungsfall *OML_0-REN_0* bei Parametereinstellung (P1) aus genau den Verfahrensvarianten aus der Häufigkeitstabelle zum 1. Platz (Tabelle 12).

in den Grundtypen (*OML_∗-REN_∗*) überein. Demnach spielen die Grundverfahren aus der kompletten Überdeckungsmenge zur Parameterdarstellung (P1) die führende Rolle bzgl. der Häufigkeitskriterien.

An dieser Stelle sei noch darauf hingewiesen, dass eine komplette Überdeckungsmenge aus weit weniger Verfahren bestehen kann, als die Liste der Verfahren mit positiven Häufigkeitswerten (TOP 1) beinhaltet, hier 13 statt 43 (vgl. Tabelle 29). Zudem sei angemerkt, dass im ausgeführten Fall die komplette Überdeckungsmenge nur 1,33% aller Verfahren ausmacht, was insbesondere im Hinblick auf Rechenzeitersparnis eine sehr entscheidende Rolle spielt.

Abschließend sollen auch die Gütekriterien zur kardinalen Messgröße *Abweichung vom besten Ergebnis (GA)* in den Ausprägungen *Durchschnittliche prozentuale Abweichung* und *Maximale prozentuale Abweichung* hinzugezogen werden. Hier zeigt sich, dass die Ausweitung auf alle 980 Verfahrensvarianten (Tabelle 32) grundsätzlich zu denselben Ergebnissen führt als zuvor im eingeschränkten Fall *OML_0-REN_0* mit 245 Grundvarianten (Tabelle 21.), zumal eine große Übereinstimmung der beiden Listen, besonders in den oberen Rängen, und eine sehr große Überschneidung mit der kompletten Überdeckungsmenge besteht, sodass deren Verfahrensvarianten durch das Gütekriterium *Durchschnittliche prozentuale Abweichung von bestem Ergebnis (GA)* ihre hervorgehobene Stellung bestätigt bekommen.

Tabelle 31: *Komplette Überdeckungsmenge beim Vergleich der durchschnittlichen Platzierung (DPl) und Prozentualer Abweichung (DPA) aller 980 Varianten zur Parametereinstellung (P1); sortiert nach ID*

ID	Verfahren	Häufigkeit TOP 1	DPl	DPA
602	SucC-LM_max-LIFO-NW-OML_1-REN_0-PO_Median	3	6,5	2,38%
842	SucC-LM_max-FIFO-KP_min-OML_1-REN_1-PO_Median	7	6	1,75%
843	SucC-LM_max-FIFO-LM_max-OML_1-REN_1-PO_Median	1	23	5,83%
844	SucC-LM_max-FIFO-NW-OML_1-REN_1-PO_Median	28	2	0,95%
845	SucC-LM_max-LIFO-KP_min-OML_1-REN_1-PO_Median	1	12,5	2,90%
846	SucC-LM_max-LIFO-LM_max-OML_1-REN_1-PO_Median	1	29,5	6,64%
850	SucC-LM_min-FIFO-NW-OML_1-REN_1-PO_Median	2	10	2,73%
854	SucC-NW-FIFO-KP_min-OML_1-REN_1-PO_Median	7	8	2,00%
855	SucC-NW-FIFO-LM_max-OML_1-REN_1-PO_Median	2	23	5,43%
856	SucC-NW-FIFO-NW-OML_1-REN_1-PO_Median	6	4,5	1,62%
857	SucC-NW-LIFO-KP_min-OML_1-REN_1-PO_Median	1	9,5	2,59%
859	SucC-NW-LIFO-NW-OML_1-REN_1-PO_Median	6	4,5	1,80%
861	SimC-NW-LM_max-Impl-OML_1-REN_1-PO_Median	1	51	7,61%

Tabelle 32: *Durchschnittliche (DPA) und maximale (MPA) prozentuale Abweichung vom besten Ergebnis (GA) beim Vergleich aller 980 Varianten zur Parametereinstellung (P1); Abweichungswerte* $\leq 2,5\%$

ID	Verfahren	DPA	MPA
844	SucC-LM_max-FIFO-NW-OML_1-REN_1-PO_Median	0,95%	7,78%
354	SucC-LM_max-FIFO-NW-OML_0-REN_1-PO_Median	1,26%	8,11%
599	SucC-LM_max-FIFO-NW-OML_1-REN_0-PO_Median	1,36%	7,87%
856	SucC-NW-FIFO-NW-OML_1-REN_1-PO_Median	1,62%	10,37%
109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median	1,66%	11,09%
842	SucC-LM_max-FIFO-KP_min-OML_1-REN_1-PO_Median	1,75%	5,74%
859	SucC-NW-LIFO-NW-OML_1-REN_1-PO_Median	1,80%	12,10%
366	SucC-NW-FIFO-NW-OML_0-REN_1-PO_Median	1,91%	10,37%
611	SucC-NW-FIFO-NW-OML_1-REN_0-PO_Median	1,99%	10,37%
854	SucC-NW-FIFO-KP_min-OML_1-REN_1-PO_Median	2,00%	6,08%
352	SucC-LM_max-FIFO-KP_min-OML_0-REN_1-PO_Median	2,01%	8,81%
847	SucC-LM_max-LIFO-NW-OML_1-REN_1-PO_Median	2,04%	10,62%
369	SucC-NW-LIFO-NW-OML_0-REN_1-PO_Median	2,05%	12,10%
597	SucC-LM_max-FIFO-KP_min-OML_1-REN_0-PO_Median	2,15%	9,47%
614	SucC-NW-LIFO-NW-OML_1-REN_0-PO_Median	2,19%	12,10%
121	SucC-NW-FIFO-NW-OML_0-REN_0-PO_Median	2,29%	11,80%
364	SucC-NW-FIFO-KP_min-OML_0-REN_1-PO_Median	2,33%	11,98%
602	SucC-LM_max-LIFO-NW-OML_1-REN_0-PO_Median	2,38%	10,62%
107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_Median	2,41%	10,93%
609	SucC-NW-FIFO-KP_min-OML_1-REN_0-PO_Median	2,42%	9,72%
124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median	2,46%	12,10%
357	SucC-LM_max-LIFO-NW-OML_0-REN_1-PO_Median	2,46%	12,48%

6.4.1.7 Zwischenfazit und Empfehlungen

Die Ergebnisse anhand der getesteten Gartenbeispiele zeigen, dass viele deterministische Verfahren in ihrer Lösungsgüte keine große Rolle spielen, und zwar hinsichtlich aller betrachteten Gütekriterien, und somit durchaus vernachlässigt werden können. Umgekehrt gibt es einige Verfahren, die bei nahezu allen Gütekriterien durchweg gut abschneiden. Diese Verfahren sind in der kompletten Überdeckungsmenge enthalten. Die komplette Überdeckungsmenge zeigt aber auch auf, dass zur tendenziellen Gewährleistung, die beste Lösung zu liefern, die anhand der implementierten deterministischen Verfahrensvarianten erreicht werden kann, auch solche Verfahren eine signifikante Rolle spielen können, die nur im Einzelfall besondere Lösungsgüte aufweisen.

Wenn es also darum geht, für einen konkreten „neuen“ Garten eine möglichst beste Harkvorschrift zu bestimmen und dabei den gesamten Rechenaufwand für alle verfügbaren Verfahrensvarianten deutlich zu verkürzen, empfiehlt es sich in der Tat, auf eine komplette Überdeckungsmenge zurückzugreifen, welche anhand

von Tests an vergleichbaren Gärten ermittelt worden ist. Zwar ist damit nicht gewährleistet, dass in jedem konkreten Fall die bestmögliche Lösung ermittelt wird, aber zumindest ist ein gutes Abschneiden zu erwarten. Diese kompletten Überdeckungsmengen bestehen je nach Parametereinstellung aus sehr wenigen Verfahrensvarianten, zumeist weniger als 2% aller verfügbaren Verfahren (vgl. Abschnitt 6.4.1.3).

Die kompletten Überdeckungsmengen bieten sich grundsätzlich auch dafür an, wenn eine Auswahl von (sehr bis einigermaßen) guten Startlösungen für die bionischen Verfahren getroffen werden soll. Diese Überdeckungsmengen liefern eine Streuung von weitgehend sehr guten bis vereinzelt guten Lösungen, was durchaus in das Konzept von bionischen Verfahren passt, einerseits einen großen, andererseits aber auch zielgerichteten Suchraum vorzugeben. Wenn aber konzeptionell auch noch zufällig erzeugte Startlösungen zur Auswahl hinzugenommen werden sollen, könnte sich dadurch die Populationsgröße zu sehr ausweiten. Daher kann es ratsam sein, nur auf eine beschränkte Überdeckungsmenge zurückzugreifen. Aber auch dann wird eine gewünschte Gütestreuung durchaus noch gegeben sein.

Aufgrund dieser Vorüberlegungen sind die Harkvorschriften der in Tabelle 33 aufgelisteten deterministischen Verfahrensvarianten als Startlösungen für die bionischen Verfahren ausgewählt worden. Hierbei wurde auf die Varianten mit Laubmengenobergrenzenvariation verzichtet, um in diesem Vorbereitungsschritt Rechenzeiten einzusparen und potentiell vorweggenommene Verbesserungen den bionischen Verfahren zu überlassen. Für einen späteren Vergleich mit den bionischen Verfahren werden hier zudem die Rechenzeiten für eine komplette Überdeckungsmenge angegeben. Zur Ermittlung der besten Lösung unter den Verfahren aus der kompletten Überdeckungsmenge benötigt man in Abhängigkeit von der Gartengröße zwischen 1 Sekunde und ca. 3 Minuten im Schnitt. Die Entwicklung der Rechenzeiten in Abhängigkeit von der Gartengröße (gemessen an der Anzahl der Felder) ist in Abbildung 34 exemplarisch zur Parametereinstellung (P1) skizziert.

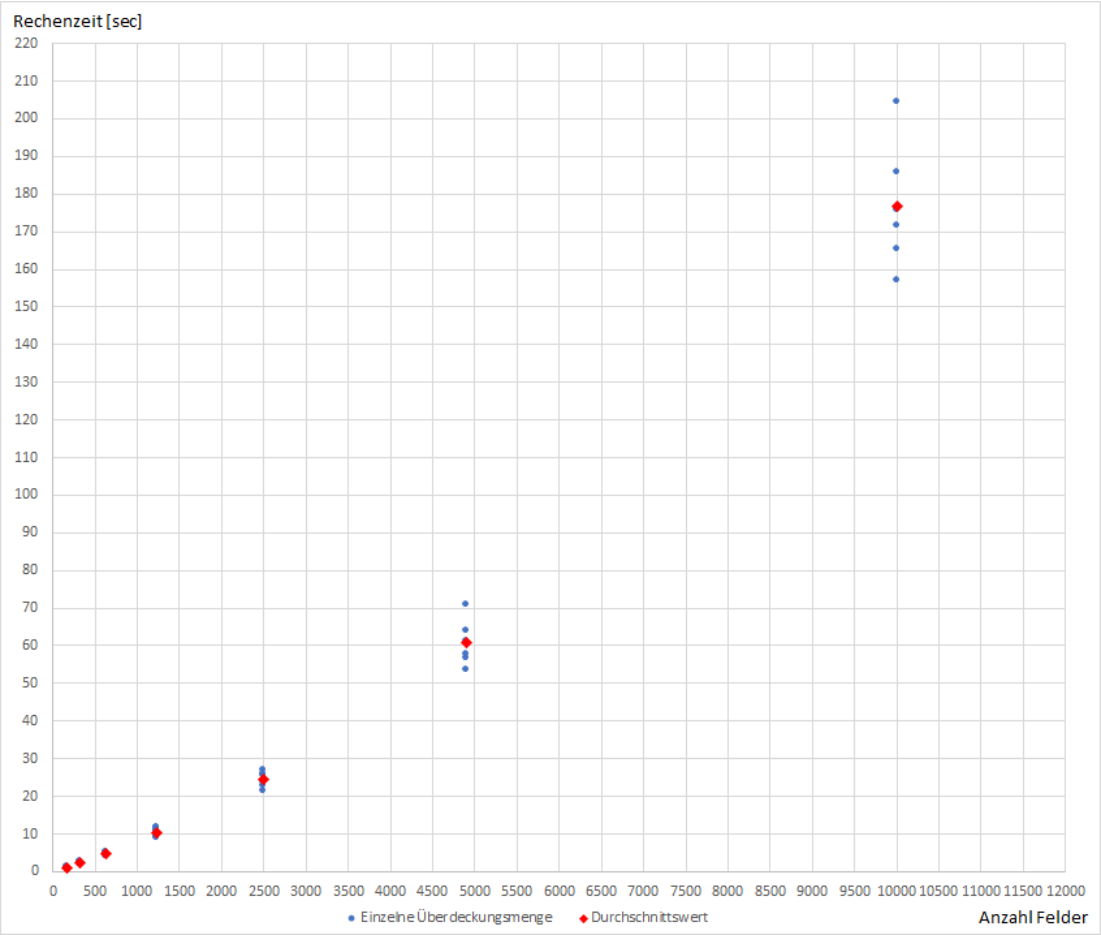


Abbildung 34: *Rechenzeiten der kompletten Überdeckungsmengen in Abhängigkeit von der Gartengröße bei Parametereinstellung (P1)*

Tabelle 33: *Ausgewählte deterministische Verfahren als Erzeuger von Startlösungen für die bionischen Verfahren*

Parametereinstellung (P1)	
354	**SucC-LM_max-FIFO-NW-OML_0-REN_1-PO_Median**
364	**SucC-NW-FIFO-KP_min-OML_0-REN_1-PO_Median**
369	**SucC-NW-LIFO-NW-OML_0-REN_1-PO_Median**
352	**SucC-LM_max-FIFO-KP_min-OML_0-REN_1-PO_Median**
366	**SucC-NW-FIFO-NW-OML_0-REN_1-PO_Median**
357	**SucC-LM_max-LIFO-NW-OML_0-REN_1-PO_Median**
360	**SucC-LM_min-FIFO-NW-OML_0-REN_1-PO_Median**
363	**SucC-LM_min-LIFO-NW-OML_0-REN_1-PO_Median**
122	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_Median
353	SucC-LM_max-FIFO-LM_max-OML_0-REN_1-PO_Median
355	SucC-LM_max-LIFO-KP_min-OML_0-REN_1-PO_Median
356	SucC-LM_max-LIFO-LM_max-OML_0-REN_1-PO_Median
365	SucC-NW-FIFO-LM_max-OML_0-REN_1-PO_Median
371	SimC-NW-LM_max-Impl-OML_0-REN_1-PO_Median
Parametereinstellung (P2)	
354	**SucC-LM_max-FIFO-NW-OML_0-REN_1-PO_Median**
364	**SucC-NW-FIFO-KP_min-OML_0-REN_1-PO_Median**
366	**SucC-NW-FIFO-NW-OML_0-REN_1-PO_Median**
369	**SucC-NW-LIFO-NW-OML_0-REN_1-PO_Median**
352	**SucC-LM_max-FIFO-KP_min-OML_0-REN_1-PO_Median**
360	**SucC-LM_min-FIFO-NW-OML_0-REN_1-PO_Median**
355	**SucC-LM_max-LIFO-KP_min-OML_0-REN_1-PO_Median**
357	**SucC-LM_max-LIFO-NW-OML_0-REN_1-PO_Median**
108	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_Median
122	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_Median
353	SucC-LM_max-FIFO-LM_max-OML_0-REN_1-PO_Median
Parametereinstellung (P3)	
354	**SucC-LM_max-FIFO-NW-OML_0-REN_1-PO_Median**
366	**SucC-NW-FIFO-NW-OML_0-REN_1-PO_Median**
364	**SucC-NW-FIFO-KP_min-OML_0-REN_1-PO_Median**
352	**SucC-LM_max-FIFO-KP_min-OML_0-REN_1-PO_Median**
369	**SucC-NW-LIFO-NW-OML_0-REN_1-PO_Median**
112	**SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median**
357	**SucC-LM_max-LIFO-NW-OML_0-REN_1-PO_Median**
355	**SucC-LM_max-LIFO-KP_min-OML_0-REN_1-PO_Median**
108	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_Median
122	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_Median
353	SucC-LM_max-FIFO-LM_max-OML_0-REN_1-PO_Median
360	SucC-LM_min-FIFO-NW-OML_0-REN_1-PO_Median

Bei einer Beschränkung der kompletten Überdeckungsmengen auf die grün hinterlegten Varianten ergibt sich je nach Parametereinstellung (P1) - (P3) eine Überdeckung der jeweils besten Lösungen für 60 bzw. 63 bzw. 62 der 66 Testgärten. Dabei liefern diese beschränkten Überdeckungsmengen eine durchschnittliche

Abweichung von 0,05% bzw. 0, 02% bzw. 0,06% im Vergleich zur jeweils kompletten Überdeckung, wobei sich die Rechenzeiten gegenüber der kompletten Überdeckung in ungefährer Entsprechung zum jeweiligen Anteil der einbezogenen Verfahren um ca. 45% bzw. 30% bzw. 35% verringern.

Die Ergebnisse, welche die bionischen Verfahren mit und ohne Hilfe dieser Startlösungen liefern, werden im folgenden Abschnitt 6.4.2 präsentiert.

6.4.2 Auswertungen zu den bionischen Verfahren

Bionische Verfahren bilden im Allgemeinen Entwicklungsabläufe und Verhaltensweisen aus der Natur ab. Entwicklungen in der Natur brauchen bekanntlich Zeit, sehr viel Zeit, wenn man dabei an die Evolution oder die Genetik denkt, insbesondere dann, wenn man von den Ursprüngen der jeweiligen Entwicklungen ausgeht. Daher ist es nicht verwunderlich, dass auch bionische Verfahren viel Zeit in Anspruch nehmen, wenn sich die Ergebnisse in Richtung von Optimallösungen entwickeln sollen, und insbesondere auch hier, wenn die Startlösungen ohne Voraussicht auf das Optimierungsziel willkürlich und einfach gesetzt werden. Um so eindrucksvoller ist es dann, dass die implementierten bionischen Verfahren auf der Grundlage von willkürlichen Startlösungen zu guten Ergebnissen führen, auch wenn das jeweilige Konvergenzverhalten bzw. die damit verbundene Rechenzeit ein gewisses Manko darstellt. Benutzt man allerdings bereits ermittelte „gute“ Startlösungen, so zeigt sich, das die bionischen Verfahren zu sehr guten Lösungen führen und somit als Verbesserungsverfahren empfohlen werden können, insbesondere dann, wenn der sekundären Zielgröße *Rechenaufwand* nur eingeschränktes Gewicht beigemessen wird.

Natürlich ist nicht *jedes* bionische Verfahren für *jede* Problemstellung als Lösungsansatz geeignet. Vielmehr muss die vorhandene Problemstruktur dem originären Naturprozess durch geeignete Problemmodellierung möglichst gut angepasst werden können. Für das hier behandelte Laubharkproblem zeigt sich im Vorgriff auf die folgenden Auswertungsergebnisse, dass sich der Bienenalgorithmus eindeutig als das geeignetere bionische Verfahren gegenüber dem genetischen Lösungsansatz herausstellt.

Zu den Testergebnissen im Einzelnen[87]:
Es werden die Lösungsergebnisse der beiden hier untersuchten bionischen Verfahren (genetischer Algorithmus (GA) und Bienenalgorithmus (BA)) einerseits als „Stand-alone-Verfahren“ betrachtet (d.h. ohne Vorgabe guter Startlösungen) und den Ergebnissen der deterministischen Verfahren gegenübergestellt, andererseits als Verbesserungsverfahren, wobei die Lösungen der deterministischen Verfahren als Startlösungen verwendet werden. Dazu wurden 162 Testszenarien zur Auswertung hinzugezogen. Hierbei handelt es sich um die kleinen und mittelgroßen Testgärten bis einschließlich der Größe 50×50, wobei zunächst noch keine Unterscheidung zwischen den Parametereinstellungen (P1) - (P3) gemacht wird.

Der direkte Vergleich der beiden bionischen Lösungsansätze ist eindeutig: Der Bienenalgorithmus erzeugt in allen untersuchten Fällen die besseren Lösungen. Im Stand-alone-Fall schneidet der Bienenalgorithmus um durchschnittlich 4, 64% besser als der genetische Algorithmus ab, vereinzelt sogar über 9%. Beim Einsatz als Verbesserungsverfahren liefert der Bienenalgorithmus im Schnitt 1% bessere Lösungen gegenüber dem genetischen Algorithmus, vereinzelt sogar über 4,6% (vgl. Abbildung 35).

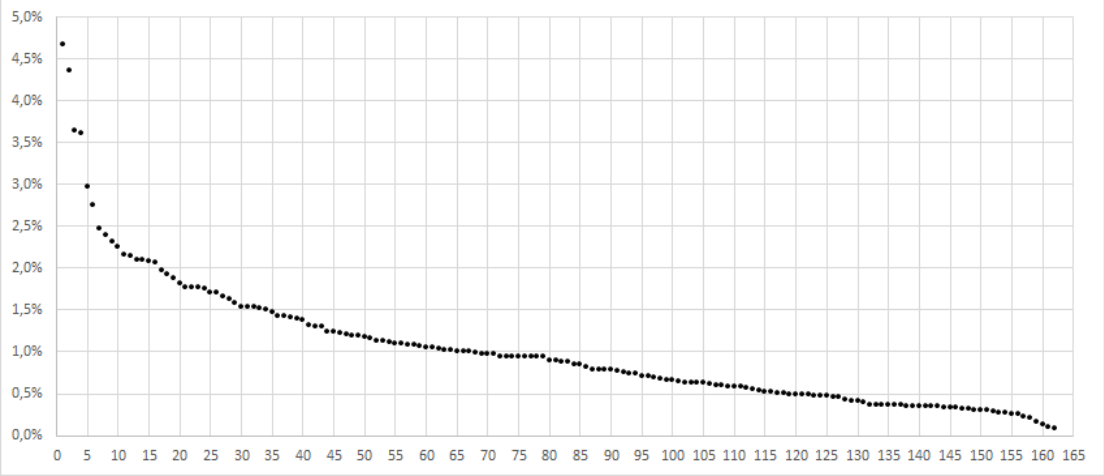


Abbildung 35: *Direkter Vergleich der bionischen Verfahren an 162 Testgärten, geordnet nach prozentualem Unterschied*

Beim Vergleich mit der jeweils besten deterministischen Lösung muss unterschieden werden, ob letztere ohne oder mit Einbeziehung des Zusatzmoduls *Laubmengenobergrenzenvariation* ermittelt worden ist, zumal die bionischen Verfahren bei

[87] An dieser Stelle sollte angemerkt werden, dass die Testergebnisse speziell implementierten Ausprägungen der bionischen Verfahren mit bestimmten Einstellungen der Verfahrensparameter entstammen, wie in den Abschnitten 4.2 und 4.3 grundlegend beschrieben und im Abschnitt 6.3 spezifiziert wird, sodass es durchaus möglich sein kann, dass sich durch Parameteroptimierungen bessere Ergebnisse erzielen lassen, weswegen die hier vollzogenen Bewertungen als Tendenzaussagen zu betrachten sind.

ihrem Einsatz als Verbesserungsverfahren auf die deterministischen Ergebnisse *ohne* Laubmengenobergrenzenvariation als Startlösungen zurückgegriffen haben (vgl. Abschnitt 6.3).

Beim Einsatz als Stand-alone-Verfahren liefert der Bienenalgorithmus in 30% bzw. 25% aller untersuchten Fälle eine bessere Lösung gegenüber dem besten deterministischen Ergebnis ohne bzw. mit Laubmengenobergrenzenvariation. In den Fällen des besseren Abschneidens beläuft sich der Unterschied vereinzelt auf über 11,8% (bzw. 7,2%), im Schnitt auf 3,25% (bzw. 3,0%) im Vergleich zur jeweils besten deterministischen Lösung. Dem stehen die Fälle des schlechteren Abschneidens mit bis zu 4,7% Unterschied und im Schnitt um 1,9% Unterschied gegenüber (bei nur geringfügigem Unterschied, ob mit oder ohne Laubmengenobergrenzenvariation). Der genetische Algorithmus liefert nur in 8% aller untersuchten Fälle eine bessere Lösung gegenüber dem besten deterministischen Ergebnis, unabhängig davon, ob die besten deterministischen Ergebnisse ohne bzw. mit Laubmengenobergrenzenvariation ermittelt worden sind. In den wenigen Fällen des besseren Abschneidens beläuft sich der Unterschied vereinzelt auf über 8,3% (bzw. 4,3%), im Schnitt auf 5,3% (bzw. 3,0%) im Vergleich zur jeweils besten deterministischen Lösung ohne bzw. mit Laubmengenobergrenzenvariation. Demgegenüber gibt es vereinzelt über 12% schlechtere Lösungen, im Schnitt ca. 6,5% (bei kaum merklichem Unterschied, ob mit oder ohne Laubmengenobergrenzenvariation).

Beim Einsatz als Verbesserungsverfahren liefert der Bienenalgorithmus in allen untersuchten Fällen verbesserte Lösungen, der genetische Algorithmus in 95% aller Fälle. Dabei schwanken die prozentualen Verbesserungen gegenüber den jeweils besten deterministischen Startlösungen (ohne Laubmengenobergrenzenvariation) beim Bienenalgorithmus zwischen 12,16% und 0,08% und liegen im Mittel bei 1,54%, beim genetischen Algorithmus zwischen 8,15% und 0%, im Mittel bei 0,55%. In Abbildung 36 ist das Verbesserungsverhalten der beiden bionischen Verfahren, geordnet nach absteigenden Prozentwerten beim Bienenalgorithmus, dargestellt.

Vergleicht man die Lösungen der bionischen Verfahren mit der jeweils besten deterministischen Lösung mit Laubmengenobergrenzenvariation, so mindert der darin vorgenommene Verbesserungsschritt die Verbesserungswerte der bionischen Verfahren leicht. Zwar liefert auch dann der Bienenalgorithmus bei allen Testszenarien (100%) die bessere Lösung, allerdings schwanken die Verbesserungswerte nun zwischen 7,52% und 0,08% und liegen im Mittel bei 1,44%. Beim genetischen Algorithmus treten in 92% der Fälle Verbesserungen auf; diese

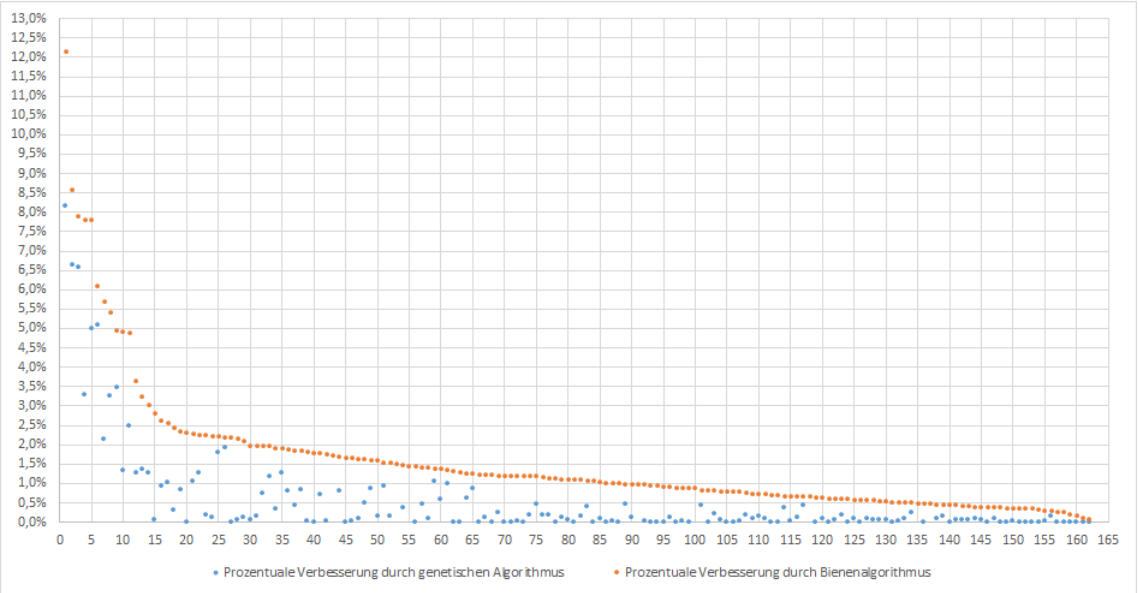


Abbildung 36: *Prozentuale Verbesserung durch genetischen und Bienenalgorithmus (absteigend sortierte Prozentwerte)*

schwanken zwischen 5,26% und 0% und liegen im Mittel bei 0,47%.

Bei den obigen Verbesserungswerten findet - wie oben erwähnt - keine Unterscheidung zwischen den einzelnen Einstellungen der Aufwandsparameter α_H, α_T und α_W statt. In Tabelle 34 sind nun die prozentualen Verbesserungen nach Parametereinstellung getrennt aufgeführt, und zwar beim Vergleich mit den besten deterministischen Lösungen ohne Laubmengenobergrenzenvariation, also zu den jeweiligen Startlösungen der bionischen Verfahren. Erwartungsgemäß nehmen die Verbesserungswerte mit steigender Gewichtung des Transportaufwandes ab (genauer gesagt: nicht zu).

Tabelle 34: *Prozentuale Verbesserungen der bionischen Verfahren im Vergleich zur jeweils besten Startlösung in Abhängigkeit von der Einstellung der Aufwandsparameter*

Einstellung	**(P1)**		**(P2)**		**(P3)**	
Verfahren	GA	BA	GA	BA	GA	BA
Maximalwert	8,15%	12,16%	1,91%	3, 65%	1,19%	2,80%
Minimalwert	0,00%	0,32%	0,00%	0, 11%	0,00%	0,08%
Durchschnittswert	1,14%	2,46%	0,25%	1, 08%	0,16%	1,08%

GA $\equiv$ Genetischer Algorithmus, BA $\equiv$ Bienenalgorithmus

Den bisherigen Vergleichen anhand der primären Ergebnisgröße (Gesamtaufwand), bei denen der Bienenalgorithmus dem genetischen Algorithmus offensichtlich überlegen ist, muss allerdings auch das Abschneiden in Bezug auf die sekundäre Ergebnisgröße (Rechenaufwand) entgegengestellt werden.

Die Rechenzeiten für den Bienenalgorithmus sind ein Vielfaches höher als für den genetischen Algorithmus. Dieser Unterschied wächst deutlich mit zunehmender Gartengröße. Die unterschiedliche Rechenzeitentwicklung in Abhängigkeit zur Gartengröße (gemessen an der Anzahl der Felder) wird exemplarisch für die Parametereinstellung (P1) in Abbildung 37 demonstriert. Dabei ist ebenfalls ersichtlich, dass die Rechenzeiten für verschiedene Testgärten derselben Größe beim Bienenalgorithmus stärker streuen als beim genetischen Algorithmus. Hierzu muss angemerkt werden, dass sich die hier zugrunde gelegten Rechenzeiten der bionischen Verfahren auf Basis von zuvor ermittelten „guten“ Startlösungen ergeben haben. Im „Stand-alone-Modus“ benötigen die bionischen Verfahren, bedingt durch eine langwierigere Konvergenzphase, zumeist sogar ein Vielfaches an Rechenzeit gegenüber dem „Verbesserungsmodus“.

Bei Zugrundelegung der beiden Ergebnisgrößen *Gesamtaufwand* und *Rechenaufwand* lässt sich ohne konkrete Präferenzangabe eines Entscheidungsträgers zwar keine endgültige Aussage über die Effizienz der bionischen Algorithmen treffen, dennoch lassen sich Trendaussagen machen: Zwar benötigt der Bienenalgorithmus bedeutend mehr Rechenzeit, ist allerdings in seinem Verbesserungsverhalten dem genetischen Algorithmus überlegen (vgl. Abbildung 36 und Tabelle 34). Daher unterliegt es einer Beurteilung durch den Entscheidungsträger, ab welcher Rechenzeitobergrenze der Einsatz des Bienenalgorithmus oder schließlich auch des genetischen Algorithmus als nicht mehr akzeptabel angesehen wird. Für kleine und mittelgroße Gärten dürfte der Bienenalgorithmus die bessere Wahl gegenüber dem genetischen Algorithmus sein. Für große Gärten könnte der genetische Algorithmus zum Zuge kommen. Allerdings sollte beachtet werden, dass beide Algorithmen mit wachsender Gartengröße eine stark abnehmende Verbesserungseffizienz aufweisen. Dieses Phänomen wird in Abschnitt 6.5 noch einer näheren Betrachtung unterzogen.

Wenn man zudem einen Vergleich mit den Rechenzeiten der deterministischen Verfahren anstellt, wird man unweigerlich zu der Beurteilung kommen, dass die bionischen Verfahren keine effizienten Alternativen als Stand-alone-Verfahren darstellen. In Tabelle 35 werden die durchschnittlichen Rechenzeiten der bionischen Algorithmen und der kompletten Überdeckungsmenge (ÜM) als ein Bündel von deterministischen Verfahren aufgelistet, exemplarisch zur Parametereinstellung (P1).

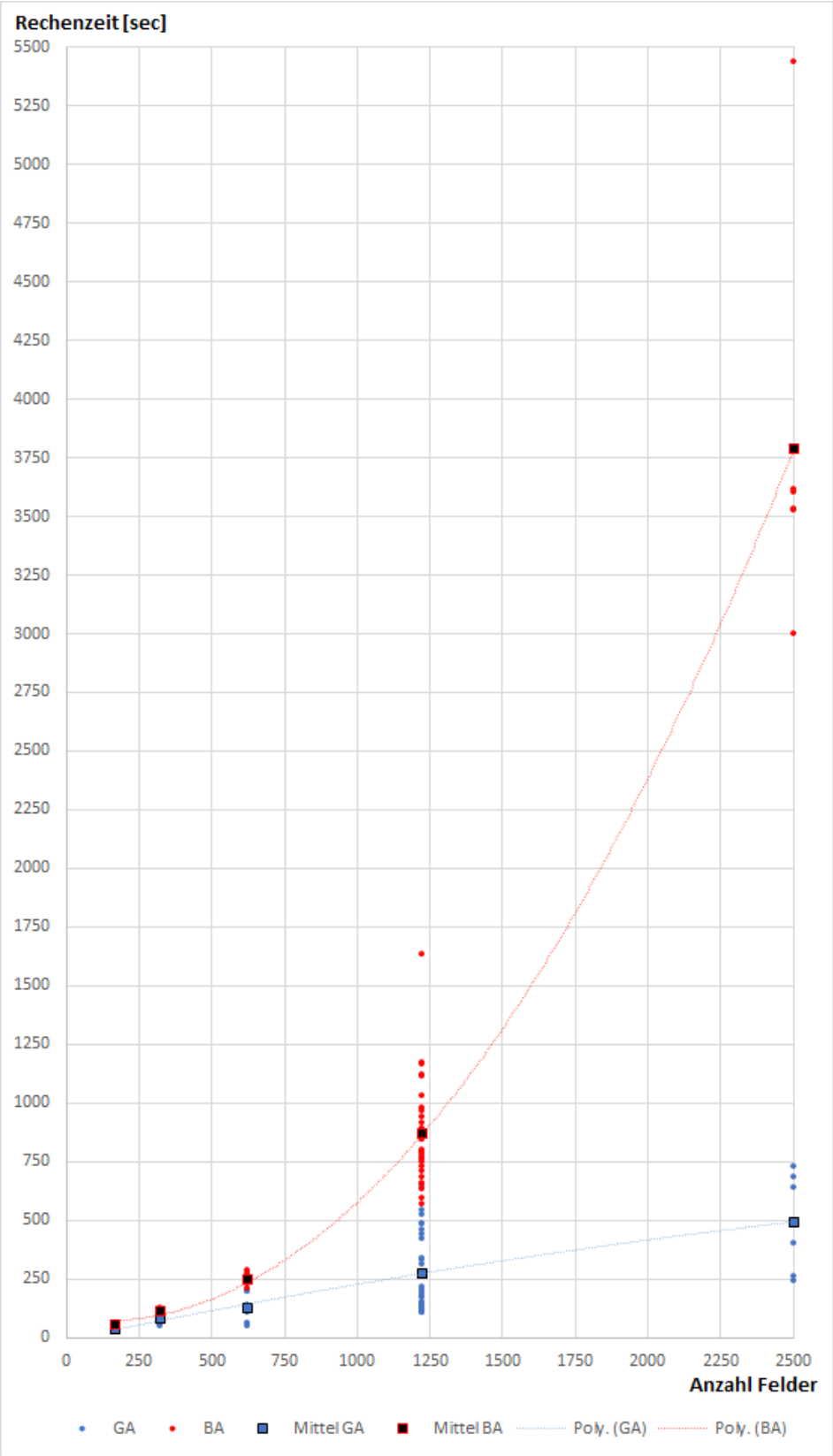


Abbildung 37: *Rechenzeiten der bionischen Verfahren in Abhängigkeit von der Gartengröße bei Parametereinstellung (P1)*

Tabelle 35: *Vergleich der durchschnittlichen Rechenzeiten der bionischen Algorithmen zur kompletten Übermenge in Abhängigkeit zur Gartengröße bei Parametereinstellung (P1)*

Gartengröße	Mittlere Rechenzeiten [sec]		
(Anzahl Felder)	**GA**	**BA**	**ÜM**
169	33,96	53,43	1,02
324	80,93	113,39	2,31
635	125,28	243,95	4,79
1225	274,86	866,97	10,47
2500	491,34	3786,34	24,37

Abschließend sei noch erwähnt, dass die Ergebnisse der bionischen Verfahren aufgrund ihrer stochastischen Komponenten in ihrer Lösungsgüte Schwankungen unterworfen sind. Um diese Streuung „abzufedern“, oder besser gesagt, um nicht „Gefahr zu laufen“, dass die Lösung nach einmaliger Durchführung zufällig einen relativ schlechten Ergebniswert aufweist, könnte man jedes bionische Verfahren mehrfach durchführen und die beste Lösung aus allen Durchläufen herausfiltern. Dieses Vorgehen aber vervielfacht die bereits hohen Rechenzeiten. Daher wurden beide bionische Verfahren anhand verschiedener Testgärten Mehrfachdurchläufen unterzogen, um die Streuung der Ergebniswerte ermitteln zu können. Es stellt sich heraus, dass die *relative Standardabweichung*[88] der Ergebniswerte dabei unter 1% liegt, was einer sehr geringen Streuung entspricht. Demnach liegen die Ergebnisse beider bionischen Verfahren in einem sehr kleinen Streubereich, sodass die Ergebnisse auch bei einmaliger Durchführung als recht zuverlässig angesehen werden können.

6.5 Zusammenfassung und kritische Bewertung der Untersuchungsergebnisse

Unter den 490 untersuchten deterministischen Verfahrensvarianten (ohne Laubmengenobergrenzenvariation) kristallisieren sich einige wenige als besonders effizient heraus. Zwecks Beschränkung auf eine sinnvolle Auswahl von möglichst wenigen Verfahrensvarianten bietet sich die komplette Überdeckungsmenge an, die - je nach Parametereinstellung - nur aus 1,9% bis 2,4% aller Verfahrensvarianten besteht (vgl. Tabelle 33) und deren Lösungen in akzeptabler Rechenzeit ermittelt werden (vgl. Abbildung 34).

[88] Die *relative Standardabweichung* (auch *Variationskoeffizient*) ist ein dimensionsloses Streuungsmaß, bei dem die Standardabweichung der Messwerte ins Verhältnis zu ihrem Mittelwert gesetzt und dadurch die Größenordnung relativiert wird; zumeist als Prozentwert angegeben.

Es ist sinnvoll und effizient, die deterministischen Verfahrensvarianten grundsätzlich mit dem Zusatzmodul *Aussparung von laubleeren Feldern* auszuführen, da sich die Lösungen dadurch in der Regel verbessern, ohne merkliche Rechenzeiterhöhungen zu verursachen.[89]

Die Hinzunahme des Zusatzmoduls *Laubmengenobergrenzenvariation* ist nur bedingt empfehlenswert, da eine zum Teil nur geringe Verbesserung der primären Ergebnisgröße *Gesamtaufwand* mit einem Vielfachen der Rechenzeit bezahlt wird.[90]

Vielmehr ist zu empfehlen, die Verbesserungen mit Hilfe der bionischen Verfahren vorzunehmen. Dabei bietet sich insbesondere der Bienenalgorithmus als Verbesserungsverfahren an, während der genetische Algorithmus als eigenständiges Lösungsverfahren nicht und als Verbesserungsverfahren nur bedingt überzeugt.[91] Auch der Bienenalgorithmus schneidet für den Stand-alone-Einsatz (d.h. ohne vorweg ermittelte gute Startlösungen) nur bedingt gut ab, auch wenn er in einzelnen Testszenarien zur jeweils besten Lösung führt, so aber aufgrund seiner vergleichsweise hohen Rechenzeiten insbesondere bei größeren Gärten keine besonders effiziente Alternative zu den besten deterministischen Verfahren bietet.

Abschließend soll noch das Phänomen angesprochen werden, dass das Verbesserungspotential (gemessen an der prozentualen Verbesserung) mit wachsender Gartengröße abnimmt und durch den gleichzeitigen Anstieg der dafür benötigten Rechenzeit zu einer kaum lohnenswerten Verbesserungseffizienz führt. Die Verringerung des Verbesserungspotentials mit wachsender Gartengröße ist in erster Linie dem Sachverhalt geschuldet, dass die Transportkosten einen immer größer werdenden Anteil an den Gesamtkosten ausmachen, bedingt durch zunehmende Entfernungen von immer mehr Hubfeldern zum Kompostfeld, weswegen effizienteres Harken und Einsparung von unproduktiven Wegen an Bedeutung verlieren. Das Laubharkproblem in seiner „1-Kompostfeld-Version“ geht also bei größer werdenden Gärten immer mehr in ein Tourenplanungsproblem über. Ab einer gewissen Gartengröße ist dann sowieso die Hinzunahme von weiteren Kompoststellen eine sinnvolle Verbesserungsmaßnahme, welche in der vorliegenden Problemstellung allerdings noch keine Betrachtung findet. Die Überlegung, das Laubharkproblem auf ein „p-Kompoststellen-Modell“ zu erweitern ($p > 1$), so wie es bei realen Ver- und Entsorgungsproblemen mit großflächigen oder überre-

[89] Näheres hierzu siehe in Abschnitt 6.4.1.5.

[90] Diese Problematik wird in Abschnitt 6.4.1.6 genauer diskutiert.

[91] Hierbei gilt der mehrfach geäußerte Hinweis, dass die bionischen Verfahren in einer speziellen Ausprägung implementiert und mit einer bestimmten Parametereinstellung versehen worden sind, worin möglicherweise noch Optimierungspotential steckt.

gionalen Gegebenheiten durchaus zur Debatte stehen könnte, führt zu einem speziellen Hubproblem, wobei die Hubs in diesem Falle die verschiedenen Endlagerstellen sind, die möglichst passend zu positionieren sind.[92] Als abschließendes Resümee bleibt festzuhalten, dass die hier vorgestellten Verfahren effiziente Lösungsansätze für Laubharkprobleme bei kleinen und mittelgroßen Gärten (bis 2500 Felder) und bedingt für große Gärten (bis 10000 Felder) darstellen, während sich bei „übergroßen“ Gärten (> 10000 Felder) die Grundsatzfrage stellt, ob das hier zugundegelegte Laubharkmodell überhaupt der passende Problemlösungsansatz ist.

[92] Diese Überlegung wird auch noch im Ausblick (Kapitel 7) aufgegriffen.

7 Ausblick

Wie an mehreren Stellen bereits angemerkt, unterliegt das hier untersuchte Laubharkmodell einer Reihe von bestimmten Vorgaben und Annahmen.[93] Insofern lässt sich auch nicht von dem Laubharkproblem schlechthin sprechen, vielmehr lassen sich eine Vielzahl von Modellmodifikationen und -erweiterungen bilden. Somit sind die hier vorgestellten Lösungsansätze und die dazu gewonnenen Einschätzungen ihrer Lösungsgüte vornehmlich auf eine bestimmte Modellklasse von allgemeinen Laubharkproblematiken bezogen und lassen nur tendenzielle Aussagen für verallgemeinerte Probleme zu. Auch dem vorgegebenen Untersuchungsrahmen ist geschuldet, dass noch viele interessante und untersuchungswürdige Fragestellungen offen bleiben.

Im Folgenden werden einige dieser offenen Fragestellungen angesprochen und problematisiert. Dabei liegt bei der vorgenommenen Reihenfolge der aufgegriffenen Punkte keine bewusste Priorisierung zugrunde, vielmehr soll durch diese Auflistung unterschiedlichster Fragestellungen ein möglicher Aufgabenkatalog für eine interessierte Leserschaft zwecks weiterer Untersuchungen angeboten werden.

(1) Modifikationen des Transportprozesses

Die Clusterbildungsprinzipien der vorgestellten deterministischen Heuristiken zielen auf eine möglichst große Laubanhäufung in den Clustern ab (möglichst nahe bei oder gleich der maximalen Laubmengenobergrenze $\overline{M}$), um durch eine möglichst geringe Clusteranzahl die Kosten für den Abtransport von den Hubs zur Kompoststelle möglichst gering zu halten. Diese Vorgehensweise ist sicherlich vorteilhaft, solange der Abtransport ausschließlich anhand einfacher Pendeltouren und die Gleichsetzung von Transportmengen- und Laubmengenobergrenze ($\overline{T} = \overline{M}$) vorgenommen wird. Bei der Kostenmodellierung wird bereits darauf verwiesen[94], dass sich durchaus auch Sammeltouren zur weiteren Verringerung der Transportkosten anbieten, insbesondere dann, wenn durch eine Harkvorschrift viele Hubs mit wenig Laubansammlung erzeugt werden. Dabei ist dann die Effi-

[93] Hierzu siehe Kapitel 2.
[94] Hierzu siehe Abschnitt 2.2.4.

zienz des zusätzlichen Einsatzes von Optimierungsverfahren für Rundreise- und Tourenplanungsprobleme[95] zu prüfen.

Unter diesem Aspekt stellt sich die grundsätzliche Frage, ob und inwieweit es sich im Hinblick auf die Gesamtkosten lohnen könnte, bei der Clusterbildung nicht mehr gezielt auf eine maximale Laubmengenanhäufung zu achten, sondern mehr „kleinere“ Cluster zu generieren, wenn durch die Ermittlung optimaler Sammeltouren die Transportkosten derart eingegrenzt werden könnten, dass sich durch die gleichzeitigen Einsparungen der Hark- und Wegekosten (kleinere Cluster bedeuten weniger Harken) eine Verringerung der Gesamtkosten einstellt, zumal neben der Verringerung der Cluster- bzw. Hubanzahl auch die Platzierung der Hubs in den Clustern eine Einsparung bei den Transportkosten ermöglicht, steuerbar über die implementierten Methoden der expliziten Hubbestimmung. Es stellt sich damit die Aufgabe, nach einem Break-even-point zur optimalen Ausbalancierung von Hark-, Wege- und Transportkosten zu suchen. Zwar ist bereits ein erster Ansatz mit der implementierten Methode der Laubmengenobergrenzenvariation vorhanden; dieses ließe sich aber im Zusammenspiel mit geeigneten Methoden zur Tourenoptimierung sicherlich noch verfeinern.

Als eine weitere Frage im Zusammenhang mit Modifikationen am Transportprozess stellt sich, ob und inwieweit die Lösungsgüte der einzelnen Verfahren von einer Änderung der Kostenstruktur für den Transportaufwand beeinflusst wird, indem beispielsweise die einfach gehaltene Transportkostenrechnung gemäß Kostenformel (2.21) durch eine detaillierte Aufteilung des Transports in einen laubmengenabhängigen Auf- und Abladeprozess und einen laubmengenunabhängigen An- und Abfahrtprozess vorgenommen wird, wie ansatzweise in Formel (2.22) angedeutet wird.

(2) Entwicklung von schnelleren Verbesserungsverfahren

Der Einsatz der untersuchten bionischen Verfahren (genetischer Algorithmus, Bienenalgorithmus) als Verbesserungsverfahren erweist sich als erfolgreich, solange die Rechenzeit keine entscheidende Rolle spielt, sondern allein die Optimierung der primären Ergebnisgröße *Gesamtaufwand* im Fokus der Untersuchung steht. Dennoch stellt sich die Frage, ob sich nicht auch „schnellere“ Verbesserungsverfahren entwickeln lassen. Dazu bieten sich grundsätzlich zwei Wege für weitere Untersuchungen an. Zum einen kann der Einsatz von bionischen Verfahren weiterverfolgt werden, wobei dann allerdings der Fokus auf eine deutliche Verringerung der Rechenzeiten gerichtet werden müsste. Bei den bereits implementierten Verfahren (genetischer Algorithmus, Bienenalgorithmus) könnte dies

[95] Näheres hierzu siehe u.a. in [6].

durch Optimierung der jeweiligen Parametereinstellungen oder durch Einsatzvariation interner Methoden versucht werden. Auch könnten andere bionische Verfahren aus dem überaus großen Angebot ausprobiert werden.[96] Zum anderen könnte auch der bereits begonnene Weg zur Entwicklung von deterministischen Verbesserungsverfahren wieder aufgenommen werden[97], der allerdings zwischenzeitlich verlassen wurde, ebenfalls aufgrund von Rechenzeitproblemen und hoffnungsfroh auf deren Ersatz durch bionische Verfahren nach dem Motto „Wenn schon, denn schon".

(3) Typisierung von Gärten und typenspezifische Verfahrenszuweisung
Anhand der untersuchten Auswahl von Beispielgärten lassen sich noch keine klaren Aussagen darüber machen, welche Verfahren für welchen Gartentyp besonders geeignet sein könnten. Zwar dominieren einige Verfahren unter bestimmten Parametersetzungen und hinsichtlich bestimmter Gütekriterien andere Verfahren über die gesamte Breite der in Augenschein genommenen Testgärten, allerdings lassen sich noch keine Anhaltspunkte für eine dazu passende Typisierung von Gärten herleiten. Letzteres ist sicherlich auch der Situation geschuldet, dass es sich bei den getesteten Gärten zwar um durchaus unterschiedliche Gärten handelt, wenn man unter anderem ihre Größen, Formen und Laubbelegungen in Dichte und Anordnung betrachtet, aber die jeweilige Anzahl in den sich dadurch ergebenden Klassen noch zu sehr gering ausfällt. Um eine passende Klassifizierung der Gärten zum Zwecke einer Zuordnung zu besonders geeigneten Verfahren vornehmen zu können, bedarf es weiterer Untersuchungen an noch mehr und ggf. anders strukturierten Gärten, wobei durchaus auch „exotische" Gartenstrukturen ins Auge zu fassen wären. Für Untersuchungen mit der Zielsetzung *Typisierung der Gärten* und *typenspezifische Eignungssuche unter den Verfahren* könnten Methoden des Data-Mining und der künstlichen Intelligenz (KI) geeignet sein. Und vielleicht fühlen sich KI-Experten auf der Suche nach weiteren Anwendungsbereichen hierdurch aufgerufen, diese offene Fragestellung der Laubharkproblematik aufzugreifen.

(4) Erweiterung des Laubharkproblems auf p Kompoststellen ($p > 1$)
Das vorgestellte Laubharkproblem mit genau einer Kompoststelle als „Endlager" könnte in interessanter Weise auch dahingehend erweitert werden, dass mehrere Kompoststellen vorliegen (Anzahl $p > 1$), die darüber hinaus in ihren Aufnahmekapazitäten auch noch beschränkt sein mögen. Hier stellt sich dann in einer zusätzlichen Lösungsstufe das Problem, eine optimale Zuordnung der Hubs zu den p (kapazitierten) Kompoststellen zu bestimmen. Zudem kann auch noch die

[96] Hierzu siehe u.a. [22].
[97] Erste Ansätze dazu finden sich in [14].

Freiheit bestehen, für p Kompoststellen zunächst geeignete Standorte zu ermitteln, was isoliert betrachtet einem (kapazitierten) p-Median-Problem[98] entspricht. Allerdings ist dabei zu beachten, dass bei mehrstufigen Optimierungsproblemen die Optimallösungen in den einzelnen Problemstufen nicht notwendigerweise direkt auch die Optimallösung für das übergeordnete Hauptproblem liefern.

(5) Verallgemeinerung von Laubhark- zum Standortmodell
Beim bisherigen Laubharkproblem geht man - geleitet durch die Anschauung von „natürlichen“ Gärten - von einem Nachbarschaftsbegriff aus, der von unmittelbar aneinander grenzenden Feldern ausgeht (hierzu siehe Abschnitt 2.1). Nun lässt sich dieses Gartenmodell natürlich auch auf allgemeinere Sammel- und Entsorgungsmodelle verallgemeinern, indem der im Modell zugrunde gelegte Graph G beliebige (direkte) Entfernungen zwischen Standorten oder Kunden repräsentiert, die ver- bzw. entsorgt werden sollen. Hierzu wird lediglich die Adjazenzmatrix und die Kantenbewertungsfunktion c entsprechend modifiziert.[99] Dabei stellt sich dann aber die Frage, ob und inwieweit sich dadurch die Lösungsgüte der einzelnen Verfahren verändert. Auch wäre zu prüfen, ob es empfehlenswert ist, die bisherigen Clusterbildungsprinzipien beizubehalten, oder ob angepasste Methoden zur Clusterbildung entwickelt werden müssen.

Es ist ganz im Sinne des Autorenteams, wenn der eine oder andere Interessierte sich die eine oder andere Fragestellung zu eigen machte und weitergehende Untersuchungen anstellte. Für Hinweise, Anregungen und Nachfragen steht das Autorenteam gerne zur Verfügung (siehe Hinweise zu den Autoren auf S. 162ff)

[98] Hierzu siehe [6].
[99] Zum Graphenmodell siehe Abschnitt 2.1, Seite 10, und Abschnitt 2.3, Seite 27.

Literaturverzeichnis

[1] Alumur, S. ; Kara, B. Y.: Network hub location problems: The state of the art. In: *European Journal of Operational Research* 190 (2008), oct, Nr. 1, 1-21. http://dx.doi.org/10.1016/j.ejor.2007.06.008. – DOI 10.1016/j.ejor.2007.06.008. – ISSN 0377–2217

[2] Bentley, P. J. ; Corne, D. W.: Creative Evolutionary Systems. In: *The Morgan Kaufmann Series in Artificial Intelligence* (2002)

[3] Breedam, A. V.: Comparing descent heuristics and metaheuristics for the vehicle routing problem. In: *Computers and Operations Research* 28 (2001), apr, Nr. 4, S. 289–315. http://dx.doi.org/10.1016/s0305-0548(99)00101-x. – DOI 10.1016/s0305–0548(99)00101–x

[4] Campbell, J. F. ; O'Kelly, M. E.: Twenty-Five Years of Hub Location Research. In: *Transportation Science* 46, No. 2 (2012), 153-169. https://doi.org/10.1287/trsc.1120.0410

[5] Cooper, W. W. ; Seiford, L. M. ; Tone, K. : *Data Envelopment Analysis*. Springer US, 2007. http://dx.doi.org/10.1007/978-0-387-45283-8. http://dx.doi.org/10.1007/978-0-387-45283-8

[6] Domschke, W. ; Scholl, A. : *Logistik: Rundreisen und Touren*. Oldenbourg Wissenschaftsverlag. http://dx.doi.org/doi:10.1524/9783486709971. http://dx.doi.org/doi:10.1524/9783486709971. – ISBN 9783486709971

[7] Dyckhoff, H. ; Allen, K. : Theoretische Begründung einer Effizienzanalyse mittels Data Envelopment Analysis (DEA). Version: may 1999. http://dx.doi.org/10.1007/bf03371573. In: *Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung* Bd. 51. Springer Science and Business Media LLC, may 1999. – DOI 10.1007/bf03371573, S. 411–436

[8] Farahani, R. Z. ; Hekmatfar, M. ; Arabani, A. B. ; Nikbakhsh, E. : Hub location problems: A review of models, classification, solution techniques, and applications. In: *Computers & Industrial Engineering* 64 (2013), apr, Nr. 4, 1096-1109. http://dx.doi.org/10.1016/j.cie.2013.01.012. – DOI 10.1016/j.cie.2013.01.012. – ISSN 0360–8352

[9] Felker, E. : Analyse der unproduktiven Wegezeiten beim Laubharkproblem. In: *Bachelorarbeit im Studiengang Angewandte Mathematik der Fachhochschule Bielefeld* (2016)

[10] Gerdes, I. ; Klawonn, F. ; Kruse, R. : *Evolutionäre Algorithmen*. Vieweg+Teubner Verlag https://www.ebook.de/de/product/33675408/ingrid_gerdes_frank_klawonn_rudolf_kruse_evolutionaere_algorithmen.html. – ISBN 9783322868398

[11] Klose, A. ; Drexl, A. : Facility location models for distribution system design. In: *European Journal of Operational Research* 162 (2005), apr, Nr. 1, 4-29. http://dx.doi.org/10.1016/j.ejor.2003.10.031. – DOI 10.1016/j.ejor.2003.10.031. – ISSN 0377–2217. – Logistics: From Theory to Application

[12] Korte, B. ; Vygen, J. : Bin-Packing. Version: 2000. http://dx.doi.org/10.1007/978-3-662-21708-5. In: *Algorithms and Combinatorics*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2000. – DOI 10.1007/978–3–662–21708–5. – ISBN 978–3–662–21708–5, S. 407–422

[13] Kruse, H.-J. : Mathematische Modellierung des Laubharkproblems als spezielle Ausprägung von allgemeinen Entsorgungsprozessen. In: Kruse, H.-J. (Hrsg.) ; Lask, T. (Hrsg.): *Angewandte mathematische Modellierung und Optimierung - Ausgewählte Modelle, Methoden, Fallstudien. Forschungsreihe des Fachbereichs Ingenieurwissenschaften und Mathematik,* Bd. 4. Fachhochschule Bielefeld, 2016. – ISSN 2196–6192, S. 62–88

[14] Kruse, H.-J. ; Spenst, N. ; Derdau, R. : Heuristische Lösungsstrategien für das Laubharkproblem. In: Kruse, H.-J. (Hrsg.) ; Lask, T. (Hrsg.): *Angewandte mathematische Modellierung und Optimierung - Ausgewählte Modelle, Methoden, Fallstudien. Forschungsreihe des Fachbereichs Ingenieurwissenschaften und Mathematik,* Bd. 4. Fachhochschule Bielefeld, 2016. – ISSN 2196–6192, S. 89–116

[15] Pham, D. T. ; Castellani, M. : The Bees Algorithm: Modelling foraging behaviour to solve continuous optimization problems. In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 223 (2009), jul, Nr. 12, S. 2919–2938. http://dx.doi.org/10.1243/09544062jmes1494. – DOI 10.1243/09544062jmes1494

[16] Savelsbergh, M. W. P.: The Vehicle Routing Problem with Time Windows: Minimizing Route Duration. In: *ORSA Journal on Computing* 4 (1992), may, Nr. 2, S. 146–154. http://dx.doi.org/10.1287/ijoc.4.2.146. – DOI 10.1287/ijoc.4.2.146

[17] Schöneburg, E. ; Heinzmann, F. ; Feddersen, S. : *Genetische Algorithmen und Evolutionsstrategien: eine Einführung in Theorie und Praxis der simulierten Evolution*. 1. Aufl., 2., unveränd. Nachdr. Bonn [u.a.] : Addison-Wesley, 1996. – ISBN 3893194932

[18] Spenst, N. : Entwicklung und Implementierung von effizienten Lösungsverfahren für ein spezielles Entsorgungsproblem. In: *Masterarbeit im Studiengang Angewandte Mathematik der Fachhochschule Bielefeld* (2015)

[19] Stepan, A. ; Fischer, E. O.: Effizienzmessung mittels Randproduktionsfunktionen und linearer Programmierung - Data Envelopment Analysis (DEA). In: *Betriebswirtschaftliche Optimierung, Einführung in die quantitative Betriebswirtschaftslehre*. Oldenbourg Wissenschaftsverlag, 2001. – ISBN 9783486256079, S. 293

[20] Turing, A. M.: I.—Computing Machinery and Intelligence. In: *Mind* LIX (1950), 10, Nr. 236, 433-460. http://dx.doi.org/10.1093/mind/lix.236.433. – DOI 10.1093/mind/lix.236.433. – ISSN 0026–4423

[21] Ulukan, Z. ; Demircioğlu, E. : A Survey of Discrete Facility Location Problems. In: *World Academy of Science, Engineering and Technology, International Journal of Industrial and Manufacturing Engineering* 9 (2015), S. 2487–2492

[22] Yang, X.-S. : *Nature-Inspired Metaheuristic Algorithms Second Edition*. Luniver Press. – 160 S. https://books.google.de/books?id=iVB_ETlh4ogC. – ISBN 9781905986286

[23] Zabihi, A. ; Gharakhani, M. : A literature survey of hub location problems and methods with emphasis on the marine transportations. In: *Uncertain Supply Chain Management* (2018), S. 91–116. http://dx.doi.org/10.5267/j.uscm.2017.5.003. – DOI 10.5267/j.uscm.2017.5.003

A Anhang

A.1 Hinweise zur Effizienzbewertung der Verfahren

Bei der Auswertung der Vergleichstests ist das Augenmerk primär auf die *Lösungsgüte* ausgerichtet, d.h. auf den Gesamtaufwand (GA) für die Laubentsorgung, welcher sich als Zielfunktionswert der durch das jeweilige Verfahren ermittelten Lösung des zugrundeliegenden Laubharkproblems ergibt. Zudem ist allerdings auch der jeweilige Rechenaufwand (Rechenzeit) als sekundäres Gütemaß von Interesse. Diese beiden Ergebnisgrößen (Zielfunktionswert, Rechenzeit) treten in der Regel als konkurrierende Messgrößen auf, d.h. für eine Verringerung des Zielfunktionswertes wird gemäß Wirtschaftlichkeits- oder Effizienz-Prinzip mehr Rechenzeit benötigt. Dabei stellt sich die Frage, für eine wie geringe Verbesserung des Zielfunktionswertes ein wie großer Rechenaufwand „lohnenswert" ist. Diese Fragestellung führt unweigerlich zum Begriff der *Effizienz* und beim Verfahrensvergleich nach dem jeweiligen *Effizienzwert* der einzelnen Verfahren.

Hierzu ein fiktives Beispiel:
Für einen Garten wurden 15 verschiedene Lösungsverfahren eingesetzt. Dabei ergaben sich die Zielfunktionswerte (GA) und Rechenzeiten (RZ) aus der folgenden Tabelle.[100] Anhand der grafischen Darstellung (Abbildung 38) wird besonders deutlich, dass von den 15 Ergebnissen 11 Ergebnisse *effizient* (im Sinne der *Pareto-Optimalität*) und davon 8 *wesentlich effizient* sind, d.h. auf der konvexen Hülle liegend. Somit lassen sich zunächst einmal die ineffizienten und ggf. auch die nicht wesentlich effizienten Ergebnisse aussondern.[101] Da bei dieser Vorgehensweise die beiden Größen *Zielfunktionswert* und *Rechenzeit* als gleichermaßen wichtig angenommen werden, lässt sich dennoch die obige Frage nach einer lohnenswerten Lösungsgüte nur sehr eingeschränkt beantworten.

[100] Hierbei wurde der Übersichtlichkeit halber und o.B.d.A. eine lexikografische Anordnung vorgenommen.

[101] In Abbildung 38 sind die wesentlich effizienten Ergebnisse grün, die zwar effizienten, aber nicht wesentlich effizienten Ergebnisse orange und die ineffizienten Ergebnisse rot markiert.

Verfahren V_i	Zielfunktions-wert (GA)	Rechenzeit (RZ)
V_0	111	2
V_1	110	3
V_2	109,5	4
V_3	109	5
V_4	109	6
V_5	108	5
V_6	108	7
V_7	107,5	7
V_8	107	8
V_9	106	9
V_{10}	106	11
V_{11}	105,5	10
V_{12}	105	12
V_{13}	104,75	20
V_{14}	104,5	100

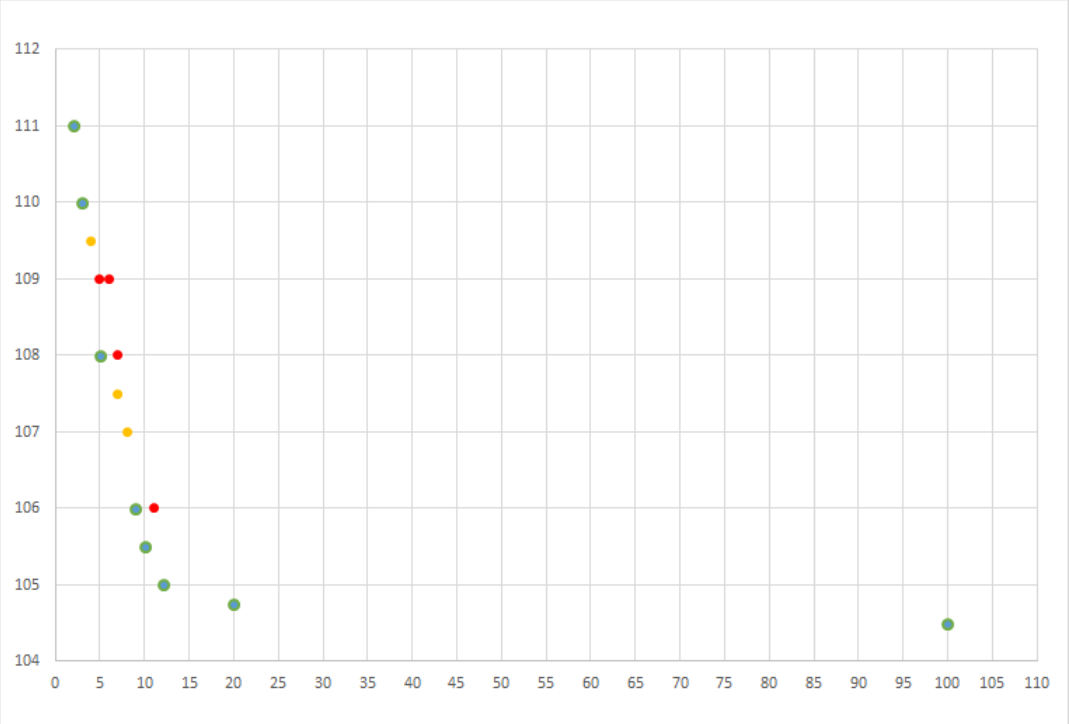


Abbildung 38: *Ergebnisvergleich bei einem fiktiven Gartenbeispiel*

Zur weiterführenden Effizienzermittlung wird auf eine gängige Methodik zurückgegriffen: *Data Envelopment Analysis (DEA)* ist eine Methode zur Messung von Effizienz für einzelne „Entscheidungseinheiten“ (*Decision Making Units*; DMU).[102]

[102]Näheres dazu siehe [19], [7], [5].

Beispielsweise werden in einem Produktionsprozess die Transformationen von eingesetzten Produktionsfaktoren (*Inputs*) in hergestellte Produkte (*Outputs*) gemessen, indem die *Produktivität* der einzelnen DMU ermittelt und vergleichend gegenübergestellt werden:

$$Produktivität = \frac{Output}{Input} \text{ (z.B. } = \frac{Ertrag}{Aufwand}).$$

Eine Effizienzanalyse im Sinne eines Produktivitätsvergleichs wird in der Praxis sehr oft über das Verhältnis zwischen *tatsächlicher Produktivität* und *höchster beobachteter Produktivität* (*relative Effizienz*) vollzogen:

$$Relative\ Effizienz = \frac{tatsächliche\ Produktivität}{höchste\ beobachtete\ Produktivität}.$$

Die relative Effizienz nimmt demnach Werte zwischen 0 und 1 an. Hierbei ist der Output eine zu maximierende Größe (je mehr Output bei konstantem Input, umso besser die Produktivität bzw. Effizienz), während der Input eine zu minimierende Größe darstellt (je weniger Input bei konstantem Output, umso besser).

Im vorliegenden Fall lassen sich die einzelnen Lösungsverfahren als DMU deklarieren, wobei die jeweiligen Ergebnisse als Input/Output-Paare interpretiert werden:

Output $\equiv$ Zielfunktionswert (Gesamtaufwand),
Input $\equiv$ Rechenzeit.

Hierbei ist allerdings sowohl der Output als auch der Input als eine zu minimierende Größe gegeben. Daher muss zunächst eine sinnvolle Transformierung der Output-Größe in Betracht gezogen werden. Zunächst wird eine „schlechte“ Lösung (z.B. das Ergebnis des Zickzack-Verfahrens V_0 mit relativ hohem Gesamtaufwand in allerdings relativ kurzer Rechenzeit) als Ausgangspunkt gewählt (quasi als „Nullpunkt“ für weitere Betrachtungen). Die Ergebnisse der anderen Verfahren V_i (Gesamtaufwand von $V_i \equiv GA_i$ und Rechenzeit von $V_i \equiv RZ_i$) werden mit dem Zickzack-Ergebnis ($GA_0; RZ_0$) wie folgt „verrechnet“:

Output $\equiv$ Verbesserung des Gesamtaufwandes bei $V_i \equiv GA_0 - GA_i$,
Input $\equiv$ „Mehrzeit“ bei $V_i \equiv RZ_i - RZ_0$.

Für obiges Beispiel ergibt sich:

Verfahren V_i	Zielwert (GA)	Rechenzeit (RZ)	Verbesserung von GA	Mehrzeit	Produktivität	Relative Effizienz
V_0	111	2	0	0	-	-
V_1	110	3	1	1	1,00	1,00
V_2	109,5	4	1,5	2	0,75	0,75
V_3	109	5	2	3	0,67	0,67
V_4	109	6	2	4	0,50	0,50
V_5	108	5	3	3	1,00	1,00
V_6	108	7	3	5	0,60	0,60
V_7	107,5	7	3,5	5	0,70	0,70
V_8	107	8	4	6	0,67	0,67
V_9	106	9	5	7	0,71	0,71
V_{10}	106	11	5	9	0,56	0,56
V_{11}	105,5	10	5,5	8	0,69	0,69
V_{12}	105	12	6	10	0,60	0,60
V_{13}	104,75	20	6,25	18	0,35	0,35
V_{14}	104,5	100	6,5	98	0,07	0,07

Im Beispiel sind die Ergebnisse der beiden Verfahren V_1 und V_5 sozusagen „voll effizient“. Sie liegen auf der sog. *Randproduktionsfunktion* (hierzu siehe auch die grafische Darstellung der Ergebnisse in Abbildung 39). Insbesondere erscheint das Ergebnis von Verfahren V_{14} als sehr wenig effizient (rel. Effizienz = 0,07), was folgerichtig daran liegt, das eine relativ hohe Rechenzeit für eine Verbesserung des Zielfunktionswertes um 3,5 „Aufwandseinheiten“ bzw. 3,24% gegenüber dem Ergebnis von Verfahren V_5 benötigt hat.

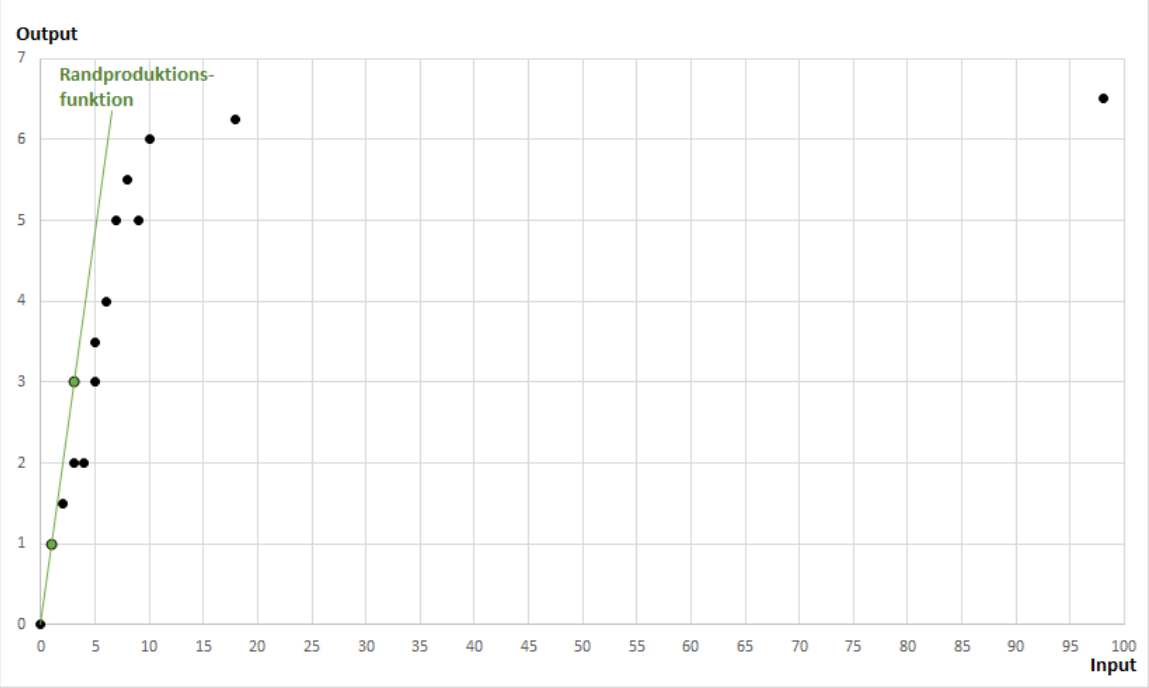


Abbildung 39: *Ergebnisdarstellung mit Randproduktionsfunktion*

Es sei angemerkt, dass die Gleichheit zwischen *Produktivität* und *relativer Effizienz* den Werten des Beispiels geschuldet und nicht die Regel ist. Zudem sei festzuhalten, dass die relative Effizienz grundsätzlich invariant gegenüber Veränderungen der Maßeinheiten ist.
Es kann angenommen werden, dass die Rechenzeit in einem gewissen Rahmen als „akzeptabel“ gelten kann, insbesondere dann, wenn selbst für eine marginale Verbesserung des Zielfunktionswertes eine hohe Rechenzeit in Kauf genommen wird. Um der „Wichtigkeit“ des Zielfunktionswertes gegenüber dem Rechenaufwand in der Effizienzbemessung auch in passender Weise Rechnung zu tragen, kann man den Input (Mehrzeit) mit Hilfe einer monoton steigenden Funktion gewichten: $f(x) = \beta \cdot x^{\alpha}$, wobei x den ursprünglichen und $f(x)$ den gewichteten Input wiedergibt. Im Folgenden wird durchgängig $\beta = 1$ gesetzt (wegen Invarianz).

Anhand der folgenden Fallunterscheidungen werden die Auswirkungen der Parametersetzung für α am obigen Beispiel verdeutlicht:

- Der Fall $\alpha = 1$ entspricht dem obigem „Standardfall“.
- Der Fall $\alpha = 0$ spiegelt die komplette Missachtung der Rechenzeit wieder, wonach sich die Effizienz allein an den Verbesserungswerten (Output) ausrichtet.

- Im Fall $\alpha = 0,5$ ergibt sich das Verfahren V_{11} als „voll effizient“; d.h. gegenüber dem „Standardfall“ ($\alpha = 1$) wird die Zielfunktion höher gewichtet, aber geringer als im „Extremfall“ ($\alpha = 0$).

Verfahren V_i	Zielwert (GA)	Rechenzeit (RZ)	Verbesserung von GA	Gew. Mehrzeit [$\alpha = 0,5$]	Produktivität	Relative Effizienz
V_0	111	2	0	0	-	-
V_1	110	3	1	1,00	1,00	0,51
V_2	109,5	4	1,5	1,41	1,06	0,55
V_3	109	5	2	1,73	1,15	0,59
V_4	109	6	2	2,00	1,00	0,51
V_5	108	5	3	1,73	1,73	0,89
V_6	108	7	3	2,24	3,34	0,69
V_7	107,5	7	3,5	2,24	1,57	0,80
V_8	107	8	4	2,45	1,63	0,84
V_9	106	9	5	2,65	1,89	0,97
V_{10}	106	11	5	3,00	1,67	0,86
V_{11}	105,5	10	5,5	2,83	1,94	1,00
V_{12}	105	12	6	3,16	1,90	0,98
V_{13}	104,75	20	6,25	4,24	1,47	0,76
V_{14}	104,5	100	6,5	9,90	0,66	0,34

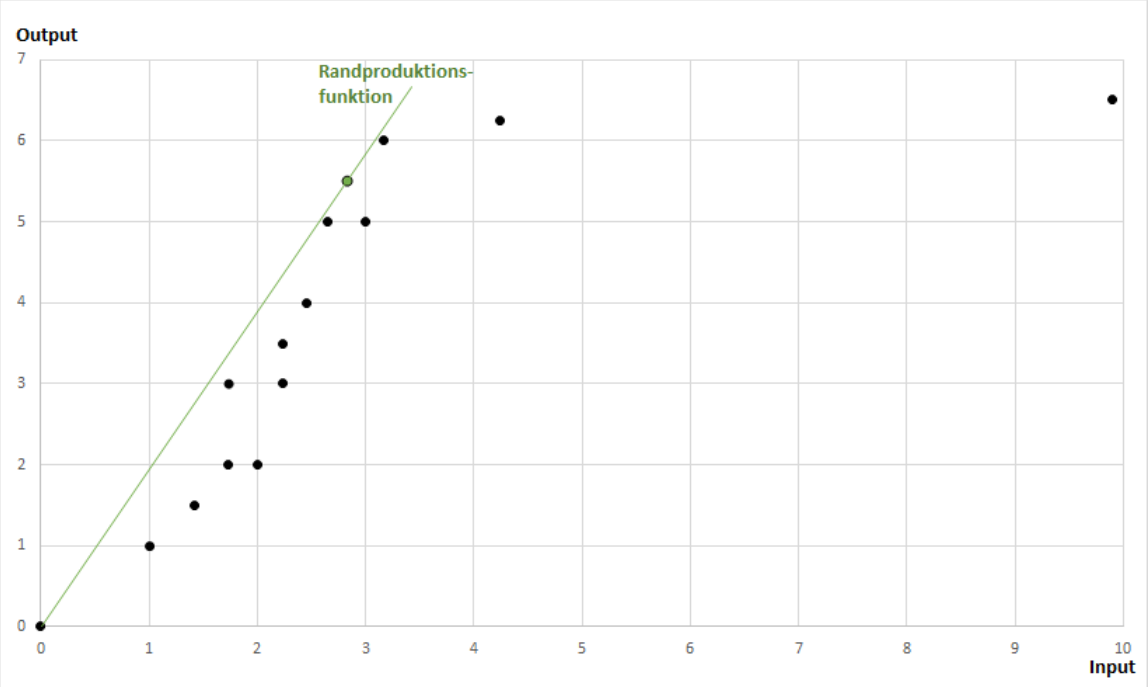


Abbildung 40: *Ergebnisdarstellung bei gewichtetem Input ($\alpha = 0,5$)*

- Im Fall $\alpha = 0,1$ verschiebt sich die Effizienzbestimmung noch weiter zugunsten der Lösungsgüte und beachtet die Rechenzeit umso weniger. Das Verfahren V_{12} gilt nun als „voll effizient“.

Verfahren V_i	Zielwert (GA)	Rechenzeit (RZ)	Verbesserung von GA	Gew. Mehrzeit [$\alpha = 0,1$]	Produktivität	Relative Effizienz
V_0	111	2	0	0	-	-
V_1	110	3	1	1,00	1,00	0,21
V_2	109,5	4	1,5	1,07	1,40	0,29
V_3	109	5	2	1,12	1,79	0,38
V_4	109	6	2	1,15	1,74	0,37
V_5	108	5	3	1,12	2,69	0,56
V_6	108	7	3	1,17	2,55	0,54
V_7	107,5	7	3,5	1,17	2,98	0,63
V_8	107	8	4	1,20	3,34	0,70
V_9	106	9	5	1,21	4,12	0,86
V_{10}	106	11	5	1,25	4,01	0,84
V_{11}	105,5	10	5,5	1,23	4,47	0,94
V_{12}	105	12	6	1,26	4,77	1,00
V_{13}	104,75	20	6,25	1,34	4,68	0,98
V_{14}	104,5	100	6,5	1,58	4,11	0,86

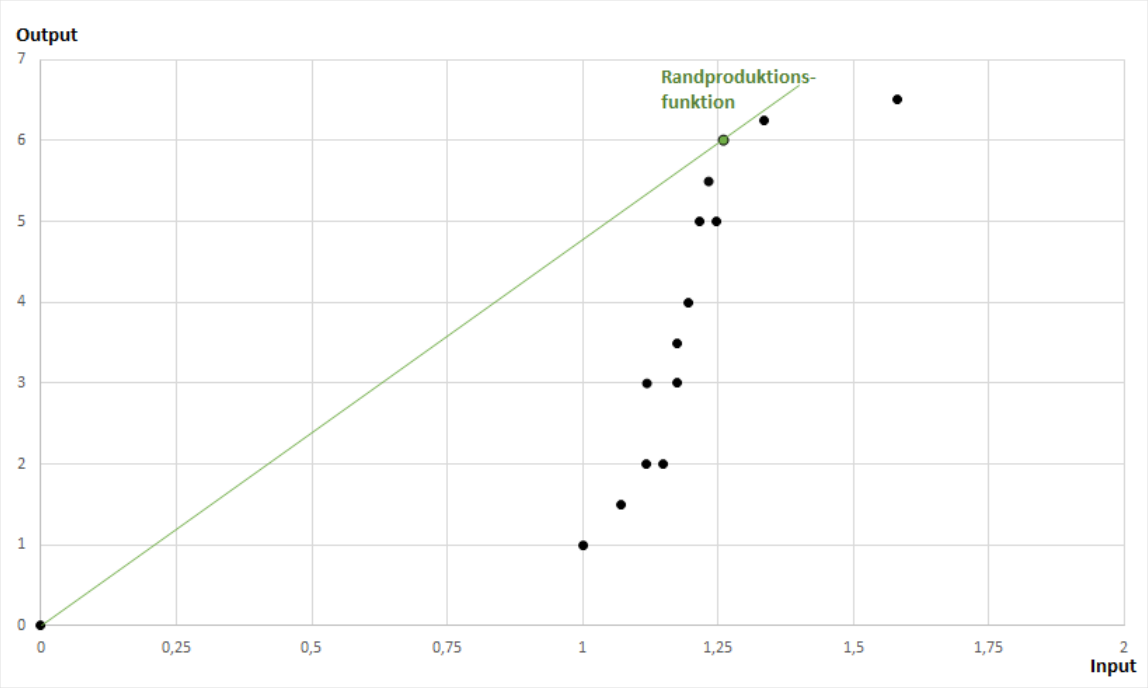


Abbildung 41: *Ergebnisdarstellung bei gewichtetem Input ($\alpha = 0,1$)*

- Bei größeren Exponenten, z.B. $\alpha = 2$, nimmt der Einfluss des Rechenaufwandes zu. Demnach ist das Verfahren V_1 „voll effizient“.

Verfahren V_i	Zielwert (GA)	Rechenzeit (RZ)	Verbesserung von GA	Gew. Mehrzeit [$\alpha = 2$]	Produktivität	Relative Effizienz
V_0	111	2	0	0	-	-
V_1	110	3	1	1	1,000	1,000
V_2	109,5	4	1,5	4	0,375	0,375
V_3	109	5	2	9	0,222	0,222
V_4	109	6	2	16	0,125	0,125
V_5	108	5	3	9	0,333	0,333
V_6	108	7	3	25	0,120	0,120
V_7	107,5	7	3,5	25	0,140	0,140
V_8	107	8	4	36	0,111	0,111
V_9	106	9	5	49	0,102	0,102
V_{10}	106	11	5	81	0,062	0,062
V_{11}	105,5	10	5,5	64	0,086	0,086
V_{12}	105	12	6	100	0,060	0,060
V_{13}	104,75	20	6,25	324	0,019	0,019
V_{14}	104,5	100	6,5	9604	0,001	0,001

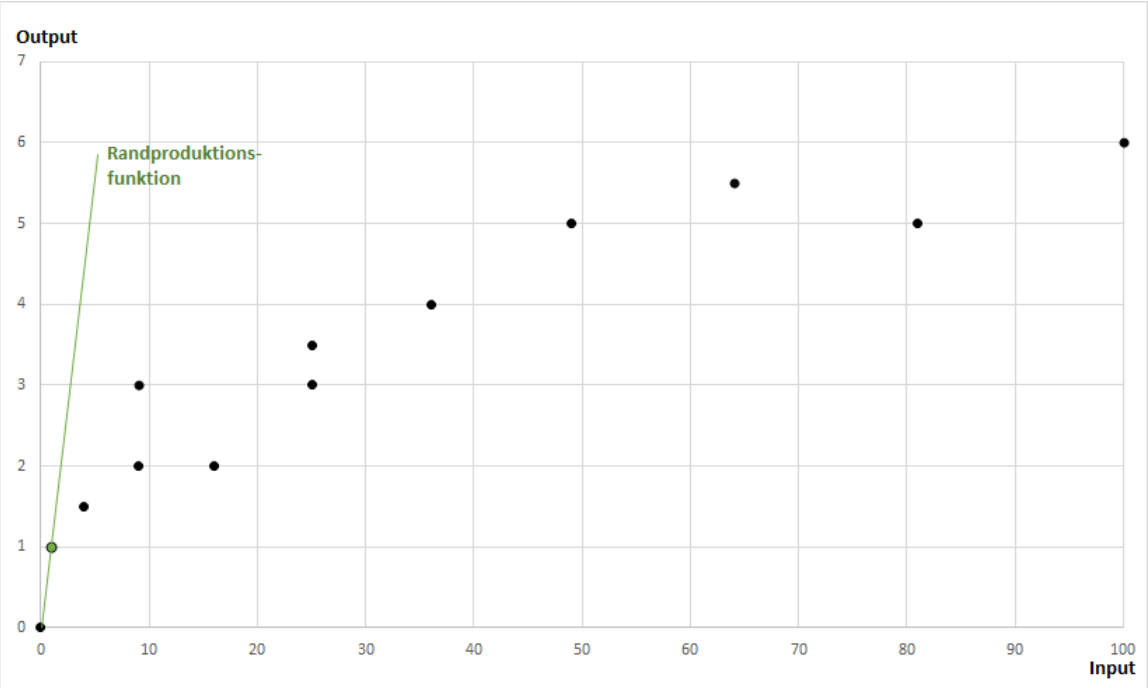


Abbildung 42: *Ergebnisdarstellung bei gewichtetem Input ($\alpha = 2$)*

An den aufgezeigten Beispielfällen wird verdeutlich, wie ein Entscheidungsträger gezielt Einfluss auf die Bedeutung der Rechenzeit bei der vorgenommenen Effizienzbewertung nehmen kann: Für Werte des Exponenten im Bereich $0 \leq \alpha < 1$ verringert sich die Bedeutung des Inputs (Mehrzeit) und damit indirekt der Rechenzeit, bis der Einfluss für $\alpha = 0$ komplett verschwindet. Im Bereich $\alpha > 1$ wird der Einfluss der Rechenzeit gegenüber dem Standardfall $\alpha = 1$ zunehmend verstärkt.

A.2 Variable Laubmengenobergrenze an einem Beispiel

Die Abhängigkeit der Lösungsgüte der deterministischen Heuristiken von der Laubmengenobergrenze soll an einem Beispiel demonstriert werden. Hierzu sei anhand der folgenden (20×20)-Matrix ein Garten mit 366 erlaubten Feldern mit den eingetragenen Laubmengen $M(a)$ und dem Kompostfeld **K** gegeben.[103] Die maximale Laubmenge pro Feld sei $\overline{M} = 50$; das maximale Ladevolumen sei $\overline{T} = 30$. Die gesamte Laubmenge im Garten beträgt ca. 1807,7 [ME], sodass jede zulässige Harkvorschrift mindestens 37 Hubs aufweisen muss.

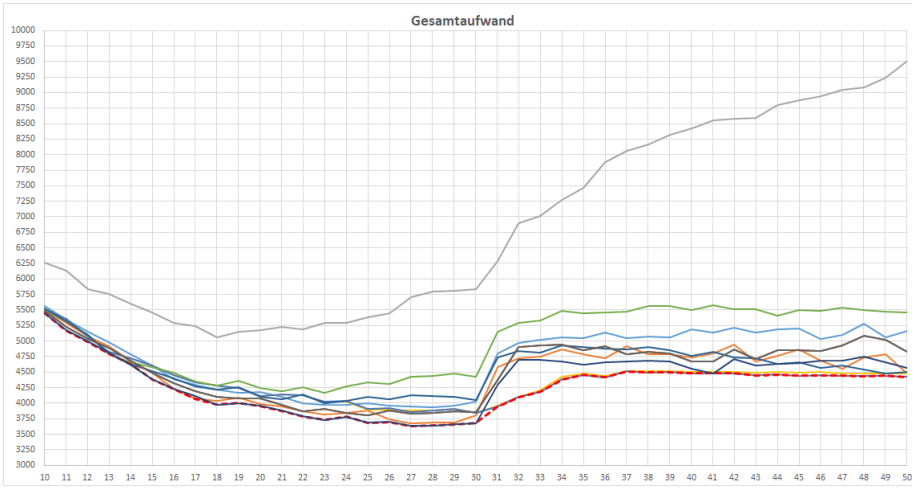
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	2,9	4,9	9,1	1,5	5,6	2,4	7,3	1,2	4,1	7,5	4	3,3	6,6	1,1	2	1,2	9,1	6,4	5,6	8,8
2	8,7	8,4	7,3	6,1	10	6,3	1,5	5,7	7,8	2,3	3,9	3,2	9,8	0,2	7,2	3,1	2	5,6	8,3	3,7
3	3	8,1	5,4	0,5	0,3	0,6	4,1	1	4,3	5,5	1,6	6,9	1,5	K	3,1	2,4	0	3,3	9,7	2,8
4	1,9	6,9	3,3	5,6	1,5	9,3	4	7,9	4,3	5,7	4,4	4,1	8,8			4,1	7,5	3,8	4,7	3,2
5	3	0,7	8	9,2	1	4,7	6,6	2								0,6	3,8	5,1	1,6	8,4
6	0,1	2,4	1,2	5,9	3,8	6,5	1,4	5,5								2,2	2,6	9,4	8,1	8,1
7	5,7	4,5	9,9	8,8	2	9,5	6,2	3,6								7,9	2	1	4,9	0,6
8	4,8	7,4	3,4	0,8	3,5	6	3,5	0,6	1,1	2,7	7,9	2,2	2	8,7	8,8	0,8	2,7	9,6	4,3	1,8
9	8,4	4	7,5	6,7	6	7,1	2,7	6,7	3,4	5,9	8	7,5	6,7	6	9,1	6,7	2,8	8,5	0,6	0,9
10	3,7	0,5	4,5	9	6,5	7,8	1,3	9,7	7,4	0,7	9,6	9,7	6,7	2,5	0,2	0,7	2,8	5,8	6,5	4,6
11	3,6	7,8	5,4	2,9	3,6	1,1	5,7	9,6	9,9	7,5	7	6,8	2,6	6,8	2,6	9,3	4,3	7,5	7,9	6,3
12	1,1	2,8	0,7	6,7	2	1,4	5,3	1,8	6,1	1,4	8,9	9,1	3,8	9,8	6,4	0,6	5,1	9,1	0,7	3,3
13	0,6	7,6	2,7	2,5	2,5			1,8	7,1	2,2	3,8	3,6	9,4	3,3	6,2	8,5	5,4	3,6	8,5	9,5
14	7,2	4,6	6,7	1,6	3,5			9,4	8,3	9,7	6,3	1,7	6	1,3	3,1	9	1	1,3	6,3	5,4
15	2,1	1,7	4,6	2,7	0,2	4,1	9,1	6,1	0,5	9,4	6,5	0,5	2,2	4,8	5,5	3,8	7,8	1,4	1,8	9,9
16	9,1	2,6	8,7	7,4	8,1	5,5	4,6	5,5	9,1	8,2	5,2	5,5	3,5	0,2	1,1	6,4	8,3	2,1	9,5	5,9
17	8,1	3,5	10	7,3	8,4	4,4	4,9	9,7	7,2	9,8	8,7		4	3		3,6	2,4	4,8	1,5	2,3
18	3	8,7	6,8	4,2	0	2,4	10	3	8,4	0,2	7,3					3,8	2,7	6,9	5,6	8,9
19	1,4	0,7	9,4	5,2	1,5	1,4	0,6	6,9	8	6,2	1,2	9,8	1,4	9,4	0,1	6,9	0,3	7,8	0,9	3,9
20	4,7	8,9	0,4	9,6	6,3	5,3	2,1	6,9	0,7	9,3	6,7	8,7	7,8	8,1	9,1	6,9	1,2	2,9	6,8	2,8

Anhand einer Auswahl von deterministischen Harkstrategien[104] wurden für verschiedene Laubmengenobergrenzen $\widetilde{M}$ die jeweilige Gesamtaufwände ermittelt, und zwar im Intervall $10 \leq \widetilde{M} \leq 50 \equiv \overline{M}$ mit der Schrittweite 1 und bei Gleichgewichtung der Aufwandsparameter ($\alpha_H = \alpha_T = \alpha_W = 1$). Die Ergebnisse sind der folgenden Tabelle bzw. der graphischen Visualisierung zu entnehmen:

[103] Die Laubmengen $M(a)$ wurden stetig gleichverteilt über dem Intervall [0;10] erzeugt, wobei die Matrixwerte nur bis zu einer Stelle hinter dem Komma angezeigt werden.

[104] Für die exemplarische Demonstration ist es unerheblich, die ausgewählten Verfahren $V_1, \ldots, V_9$ explizit zu kennen.

Gesamt	V1	V2	V3	V4	V5	V6	V7	V8	V9	MIN	% Abw
10	5560	5499	6257	5508	5507	5516	5528	5492	5451	5451	150,1%
11	5345	5289	6135	5354	5354	5315	5324	5224	5174	5174	142,5%
12	5148	5072	5834	5073	5073	5096	5100	5036	4994	4994	137,5%
13	4976	4922	5752	4828	4828	4794	4803	4893	4819	4794	132,0%
14	4793	4688	5601	4723	4723	4663	4632	4684	4623	4623	127,3%
15	4604	4486	5464	4603	4601	4577	4519	4496	4382	4382	120,7%
16	4387	4233	5290	4458	4451	4488	4399	4322	4232	4232	116,6%
17	4303	4072	5242	4344	4340	4349	4269	4199	4121	4072	112,1%
18	4222	4040	5057	4281	4286	4285	4219	4098	3978	3978	109,6%
19	4162	4088	5146	4240	4240	4359	4258	4072	4000	4000	110,2%
20	4181	3980	5170	4117	4120	4249	4101	4074	3958	3958	109,0%
21	4106	3941	5226	4141	4144	4192	4065	3973	3880	3880	106,9%
22	4005	3875	5194	4132	4127	4256	4140	3871	3791	3791	104,4%
23	3970	3817	5298	4022	4031	4165	4005	3910	3729	3729	102,7%
24	3973	3850	5289	4021	4039	4271	4042	3845	3781	3781	104,1%
25	3999	3883	5387	3890	3904	4330	4102	3805	3688	3688	101,6%
26	3957	3744	5442	3902	3916	4316	4069	3885	3697	3697	101,8%
27	3952	3679	5702	3880	3857	4422	4125	3832	3631	3631	100,0%
28	3929	3691	5792	3880	3881	4442	4121	3845	3640	3640	100,2%
29	3954	3690	5809	3911	3905	4476	4106	3873	3660	3660	100,8%
30	4027	3800	5833	3843	3848	4423	4046	3855	3680	3680	101,3%
31	4804	4586	6292	3962	3947	5149	4734	4380	4293	3947	108,7%
32	4970	4724	6891	4109	4097	5292	4838	4907	4704	4097	112,8%
33	5027	4751	7011	4206	4183	5328	4816	4924	4698	4183	115,2%
34	5060	4861	7267	4424	4386	5482	4934	4941	4677	4386	120,8%
35	5052	4782	7462	4480	4455	5447	4910	4847	4623	4455	122,7%
36	5134	4722	7884	4436	4414	5466	4875	4923	4659	4414	121,6%
37	5042	4912	8059	4512	4516	5470	4863	4789	4671	4512	124,3%
38	5079	4783	8158	4511	4496	5559	4900	4830	4685	4496	123,8%
39	5055	4782	8319	4520	4500	5560	4855	4799	4667	4500	123,9%
40	5188	4732	8418	4508	4483	5494	4760	4671	4556	4483	123,5%
41	5136	4798	8548	4504	4481	5572	4830	4678	4481	4481	123,4%
42	5216	4937	8572	4503	4483	5513	4739	4865	4713	4483	123,5%
43	5134	4669	8591	4497	4451	5518	4728	4712	4613	4451	122,6%
44	5193	4763	8793	4506	4460	5405	4639	4856	4636	4460	122,8%
45	5208	4862	8877	4493	4441	5497	4662	4859	4646	4441	122,3%
46	5038	4687	8943	4498	4447	5492	4573	4841	4685	4447	122,5%
47	5095	4555	9048	4472	4439	5536	4612	4924	4684	4439	122,3%
48	5283	4735	9076	4474	4434	5500	4547	5090	4751	4434	122,1%
49	5066	4786	9243	4483	4443	5471	4481	5019	4665	4443	122,4%
50	5159	4474	9505	4474	4426	5458	4509	4827	4569	4426	121,9%
Min	3929	3679	5057	3843	3848	4165	4005	3805	3631	3631	
% Abw	108,2%	101,3%	139,3%	105,8%	106,0%	114,7%	110,3%	104,8%	100,0%		
Rang	6	2	9	4	5	8	7	3	1		



Dabei wird für die Laubmengenobergrenze $\widetilde{M} = 27$ durch das Verfahren V_9 mit 3631 der niedrigste Gesamtaufwand erzielt. Anhand der Prozentwerte lassen sich die Abweichungen zu diesem Referenzwert ($3631 \equiv 100\%$) ablesen. Die grün unterlegten Werte zeigen das jeweils beste Ergebnis zu den verschiedenen Heuristiken in der jeweiligen Spalte an. Die jeweils besten Zielergebnisse zu den einzelnen Laubmengenobergrenzen $\widetilde{M}$ sind in der „roten Spalte“ festgehalten, welche auch als „rot gestrichelte Kurve“ in der Grafik zu erkennen ist. Die gelb

unterlegten Prozentwerte deuten auf Abweichungen von der besten Lösung mit weniger als 2% hin.

Es fällt auf, dass sich bei den verschiedenen Verfahren unterschiedliche „Bestwerte“ $\widetilde{M}_{Best}$ für die Laubmengenobergrenze $\widetilde{M}$ ergeben.

Heuristik	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
$\widetilde{M}_{Best}$	28	27	18	30	30	23	23	25	27

Allerdings ergibt sich bei der empfohlenen Setzung $\widetilde{M} = \overline{T} = 30$ gemäß (3.1) nur eine „geringe“ Abweichung von der besten Lösung. Genauer gesagt weicht die beste Lösung zur Setzung $\widetilde{M} = 30$ (Gesamtaufwand 3680) von der durch Laubmengenobergrenzenvariation ermittelten besten Lösung (Gesamtaufwand 3631) um 1,35% ab. Somit wird die empfohlene Setzung $\widetilde{M} = \overline{T}$ für die Laubmengenobergrenze bei den deterministischen Heuristiken an diesem Beispiel in gewisser Weise „bestätigt“.

Letztendlich liegt es in der Hand des Entscheidungsträgers, ob eine mögliche Lösungsverbesserung durch Laubmengenobergrenzenvariation auf Kosten der dafür benötigten Rechenzeiterhöhung vorgenommen werden soll. Das Programm `lhp` sieht diese Möglichkeit vor (hierzu siehe Kapitel 5).

A.3 Liste der deterministischen Verfahren

ID	Verfahren	ID	Verfahren
1	Zickzack-OML_0-REN_0-PO_None	71	Zickzack-OML_0-REN_0-PO_MinLaub
2	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_None	72	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_MinLaub
3	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_None	73	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_MinLaub
4	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_None	74	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_MinLaub
5	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO_None	75	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO_MinLaub
6	SucC-LM_max-LIFO-LM_max-OML_0-REN_0-PO_None	76	SucC-LM_max-LIFO-LM_max-OML_0-REN_0-PO_MinLaub
7	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_None	77	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_MinLaub
8	SucC-LM_min-FIFO-KP_min-OML_0-REN_0-PO_None	78	SucC-LM_min-FIFO-KP_min-OML_0-REN_0-PO_MinLaub
9	SucC-LM_min-FIFO-LM_max-OML_0-REN_0-PO_None	79	SucC-LM_min-FIFO-LM_max-OML_0-REN_0-PO_MinLaub
10	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_None	80	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_MinLaub
11	SucC-LM_min-LIFO-KP_min-OML_0-REN_0-PO_None	81	SucC-LM_min-LIFO-KP_min-OML_0-REN_0-PO_MinLaub
12	SucC-LM_min-LIFO-LM_max-OML_0-REN_0-PO_None	82	SucC-LM_min-LIFO-LM_max-OML_0-REN_0-PO_MinLaub
13	SucC-LM_min-LIFO-NW-OML_0-REN_0-PO_None	83	SucC-LM_min-LIFO-NW-OML_0-REN_0-PO_MinLaub
14	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_None	84	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_MinLaub
15	SucC-NW-FIFO-LM_max-OML_0-REN_0-PO_None	85	SucC-NW-FIFO-LM_max-OML_0-REN_0-PO_MinLaub
16	SucC-NW-FIFO-NW-OML_0-REN_0-PO_None	86	SucC-NW-FIFO-NW-OML_0-REN_0-PO_MinLaub
17	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_None	87	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_MinLaub
18	SucC-NW-LIFO-LM_max-OML_0-REN_0-PO_None	88	SucC-NW-LIFO-LM_max-OML_0-REN_0-PO_MinLaub
19	SucC-NW-LIFO-NW-OML_0-REN_0-PO_None	89	SucC-NW-LIFO-NW-OML_0-REN_0-PO_MinLaub
20	SimC-NW-NW-Impl-OML_0-REN_0-PO_None	90	SimC-NW-NW-Impl-OML_0-REN_0-PO_MinLaub
21	SimC-NW-LM_max-Impl-OML_0-REN_0-PO_None	91	SimC-NW-LM_max-Impl-OML_0-REN_0-PO_MinLaub
22	SimC-NW-LM_min-Impl-OML_0-REN_0-PO_None	92	SimC-NW-LM_min-Impl-OML_0-REN_0-PO_MinLaub
23	SimC-NW-KP_min-Impl-OML_0-REN_0-PO_None	93	SimC-NW-KP_min-Impl-OML_0-REN_0-PO_MinLaub
24	SimC-LM_max-NW-Impl-OML_0-REN_0-PO_None	94	SimC-LM_max-NW-Impl-OML_0-REN_0-PO_MinLaub
25	SimC-LM_max-LM_max-Impl-OML_0-REN_0-PO_None	95	SimC-LM_max-LM_max-Impl-OML_0-REN_0-PO_MinLaub
26	SimC-LM_max-LM_min-Impl-OML_0-REN_0-PO_None	96	SimC-LM_max-LM_min-Impl-OML_0-REN_0-PO_MinLaub
27	SimC-LM_max-KP_min-Impl-OML_0-REN_0-PO_None	97	SimC-LM_max-KP_min-Impl-OML_0-REN_0-PO_MinLaub
28	SimC-LM_min-NW-Impl-OML_0-REN_0-PO_None	98	SimC-LM_min-NW-Impl-OML_0-REN_0-PO_MinLaub
29	SimC-LM_min-LM_max-Impl-OML_0-REN_0-PO_None	99	SimC-LM_min-LM_max-Impl-OML_0-REN_0-PO_MinLaub
30	SimC-LM_min-LM_min-Impl-OML_0-REN_0-PO_None	100	SimC-LM_min-LM_min-Impl-OML_0-REN_0-PO_MinLaub
31	SimC-LM_min-KP_min-Impl-OML_0-REN_0-PO_None	101	SimC-LM_min-KP_min-Impl-OML_0-REN_0-PO_MinLaub
32	SimC-KP_min-NW-Impl-OML_0-REN_0-PO_None	102	SimC-KP_min-NW-Impl-OML_0-REN_0-PO_MinLaub
33	SimC-KP_min-LM_max-Impl-OML_0-REN_0-PO_None	103	SimC-KP_min-LM_max-Impl-OML_0-REN_0-PO_MinLaub
34	SimC-KP_min-LM_min-Impl-OML_0-REN_0-PO_None	104	SimC-KP_min-LM_min-Impl-OML_0-REN_0-PO_MinLaub
35	SimC-KP_min-KP_min-Impl-OML_0-REN_0-PO_None	105	SimC-KP_min-KP_min-Impl-OML_0-REN_0-PO_MinLaub
36	Zickzack-OML_0-REN_0-PO_MaxLaub	106	Zickzack-OML_0-REN_0-PO_Median
37	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_MaxLaub	107	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_Median
38	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_MaxLaub	108	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_Median
39	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_MaxLaub	109	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_Median
40	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO_MaxLaub	110	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO_Median
41	SucC-LM_max-LIFO-LM_max-OML_0-REN_0-PO_MaxLaub	111	SucC-LM_max-LIFO-LM_max-OML_0-REN_0-PO_Median
42	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_MaxLaub	112	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_Median
43	SucC-LM_min-FIFO-KP_min-OML_0-REN_0-PO_MaxLaub	113	SucC-LM_min-FIFO-KP_min-OML_0-REN_0-PO_Median
44	SucC-LM_min-FIFO-LM_max-OML_0-REN_0-PO_MaxLaub	114	SucC-LM_min-FIFO-LM_max-OML_0-REN_0-PO_Median
45	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_MaxLaub	115	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_Median
46	SucC-LM_min-LIFO-KP_min-OML_0-REN_0-PO_MaxLaub	116	SucC-LM_min-LIFO-KP_min-OML_0-REN_0-PO_Median
47	SucC-LM_min-LIFO-LM_max-OML_0-REN_0-PO_MaxLaub	117	SucC-LM_min-LIFO-LM_max-OML_0-REN_0-PO_Median
48	SucC-LM_min-LIFO-NW-OML_0-REN_0-PO_MaxLaub	118	SucC-LM_min-LIFO-NW-OML_0-REN_0-PO_Median
49	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_MaxLaub	119	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_Median
50	SucC-NW-FIFO-LM_max-OML_0-REN_0-PO_MaxLaub	120	SucC-NW-FIFO-LM_max-OML_0-REN_0-PO_Median
51	SucC-NW-FIFO-NW-OML_0-REN_0-PO_MaxLaub	121	SucC-NW-FIFO-NW-OML_0-REN_0-PO_Median
52	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_MaxLaub	122	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_Median
53	SucC-NW-LIFO-LM_max-OML_0-REN_0-PO_MaxLaub	123	SucC-NW-LIFO-LM_max-OML_0-REN_0-PO_Median
54	SucC-NW-LIFO-NW-OML_0-REN_0-PO_MaxLaub	124	SucC-NW-LIFO-NW-OML_0-REN_0-PO_Median
55	SimC-NW-NW-Impl-OML_0-REN_0-PO_MaxLaub	125	SimC-NW-NW-Impl-OML_0-REN_0-PO_Median
56	SimC-NW-LM_max-Impl-OML_0-REN_0-PO_MaxLaub	126	SimC-NW-LM_max-Impl-OML_0-REN_0-PO_Median
57	SimC-NW-LM_min-Impl-OML_0-REN_0-PO_MaxLaub	127	SimC-NW-LM_min-Impl-OML_0-REN_0-PO_Median
58	SimC-NW-KP_min-Impl-OML_0-REN_0-PO_MaxLaub	128	SimC-NW-KP_min-Impl-OML_0-REN_0-PO_Median
59	SimC-LM_max-NW-Impl-OML_0-REN_0-PO_MaxLaub	129	SimC-LM_max-NW-Impl-OML_0-REN_0-PO_Median
60	SimC-LM_max-LM_max-Impl-OML_0-REN_0-PO_MaxLaub	130	SimC-LM_max-LM_max-Impl-OML_0-REN_0-PO_Median
61	SimC-LM_max-LM_min-Impl-OML_0-REN_0-PO_MaxLaub	131	SimC-LM_max-LM_min-Impl-OML_0-REN_0-PO_Median
62	SimC-LM_max-KP_min-Impl-OML_0-REN_0-PO_MaxLaub	132	SimC-LM_max-KP_min-Impl-OML_0-REN_0-PO_Median
63	SimC-LM_min-NW-Impl-OML_0-REN_0-PO_MaxLaub	133	SimC-LM_min-NW-Impl-OML_0-REN_0-PO_Median
64	SimC-LM_min-LM_max-Impl-OML_0-REN_0-PO_MaxLaub	134	SimC-LM_min-LM_max-Impl-OML_0-REN_0-PO_Median
65	SimC-LM_min-LM_min-Impl-OML_0-REN_0-PO_MaxLaub	135	SimC-LM_min-LM_min-Impl-OML_0-REN_0-PO_Median
66	SimC-LM_min-KP_min-Impl-OML_0-REN_0-PO_MaxLaub	136	SimC-LM_min-KP_min-Impl-OML_0-REN_0-PO_Median
67	SimC-KP_min-NW-Impl-OML_0-REN_0-PO_MaxLaub	137	SimC-KP_min-NW-Impl-OML_0-REN_0-PO_Median
68	SimC-KP_min-LM_max-Impl-OML_0-REN_0-PO_MaxLaub	138	SimC-KP_min-LM_max-Impl-OML_0-REN_0-PO_Median
69	SimC-KP_min-LM_min-Impl-OML_0-REN_0-PO_MaxLaub	139	SimC-KP_min-LM_min-Impl-OML_0-REN_0-PO_Median
70	SimC-KP_min-KP_min-Impl-OML_0-REN_0-PO_MaxLaub	140	SimC-KP_min-KP_min-Impl-OML_0-REN_0-PO_Median

ID	Verfahren	ID	Verfahren
141	Zickzack-OML_0-REN_0-PO_MinKompost	211	Zickzack-OML_0-REN_0-PO_KeepHubs
142	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_MinKompost	212	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_KeepHubs
143	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_MinKompost	213	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_KeepHubs
144	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_MinKompost	214	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_KeepHubs
145	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO_MinKompost	215	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO_KeepHubs
146	SucC-LM_max-LIFO-LM_max-OML_0-REN_0-PO_MinKompost	216	SucC-LM_max-LIFO-LM_max-OML_0-REN_0-PO_KeepHubs
147	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_MinKompost	217	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_KeepHubs
148	SucC-LM_min-FIFO-KP_min-OML_0-REN_0-PO_MinKompost	218	SucC-LM_min-FIFO-KP_min-OML_0-REN_0-PO_KeepHubs
149	SucC-LM_min-FIFO-LM_max-OML_0-REN_0-PO_MinKompost	219	SucC-LM_min-FIFO-LM_max-OML_0-REN_0-PO_KeepHubs
150	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_MinKompost	220	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_KeepHubs
151	SucC-LM_min-LIFO-KP_min-OML_0-REN_0-PO_MinKompost	221	SucC-LM_min-LIFO-KP_min-OML_0-REN_0-PO_KeepHubs
152	SucC-LM_min-LIFO-LM_max-OML_0-REN_0-PO_MinKompost	222	SucC-LM_min-LIFO-LM_max-OML_0-REN_0-PO_KeepHubs
153	SucC-LM_min-LIFO-NW-OML_0-REN_0-PO_MinKompost	223	SucC-LM_min-LIFO-NW-OML_0-REN_0-PO_KeepHubs
154	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_MinKompost	224	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_KeepHubs
155	SucC-NW-FIFO-LM_max-OML_0-REN_0-PO_MinKompost	225	SucC-NW-FIFO-LM_max-OML_0-REN_0-PO_KeepHubs
156	SucC-NW-FIFO-NW-OML_0-REN_0-PO_MinKompost	226	SucC-NW-FIFO-NW-OML_0-REN_0-PO_KeepHubs
157	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_MinKompost	227	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_KeepHubs
158	SucC-NW-LIFO-LM_max-OML_0-REN_0-PO_MinKompost	228	SucC-NW-LIFO-LM_max-OML_0-REN_0-PO_KeepHubs
159	SucC-NW-LIFO-NW-OML_0-REN_0-PO_MinKompost	229	SucC-NW-LIFO-NW-OML_0-REN_0-PO_KeepHubs
160	SimC-NW-NW-Impl-OML_0-REN_0-PO_MinKompost	230	SimC-NW-NW-Impl-OML_0-REN_0-PO_KeepHubs
161	SimC-NW-LM_max-Impl-OML_0-REN_0-PO_MinKompost	231	SimC-NW-LM_max-Impl-OML_0-REN_0-PO_KeepHubs
162	SimC-NW-LM_min-Impl-OML_0-REN_0-PO_MinKompost	232	SimC-NW-LM_min-Impl-OML_0-REN_0-PO_KeepHubs
163	SimC-NW-KP_min-Impl-OML_0-REN_0-PO_MinKompost	233	SimC-NW-KP_min-Impl-OML_0-REN_0-PO_KeepHubs
164	SimC-LM_max-NW-Impl-OML_0-REN_0-PO_MinKompost	234	SimC-LM_max-NW-Impl-OML_0-REN_0-PO_KeepHubs
165	SimC-LM_max-LM_max-Impl-OML_0-REN_0-PO_MinKompost	235	SimC-LM_max-LM_max-Impl-OML_0-REN_0-PO_KeepHubs
166	SimC-LM_max-LM_min-Impl-OML_0-REN_0-PO_MinKompost	236	SimC-LM_max-LM_min-Impl-OML_0-REN_0-PO_KeepHubs
167	SimC-LM_max-KP_min-Impl-OML_0-REN_0-PO_MinKompost	237	SimC-LM_max-KP_min-Impl-OML_0-REN_0-PO_KeepHubs
168	SimC-LM_min-NW-Impl-OML_0-REN_0-PO_MinKompost	238	SimC-LM_min-NW-Impl-OML_0-REN_0-PO_KeepHubs
169	SimC-LM_min-LM_max-Impl-OML_0-REN_0-PO_MinKompost	239	SimC-LM_min-LM_max-Impl-OML_0-REN_0-PO_KeepHubs
170	SimC-LM_min-LM_min-Impl-OML_0-REN_0-PO_MinKompost	240	SimC-LM_min-LM_min-Impl-OML_0-REN_0-PO_KeepHubs
171	SimC-LM_min-KP_min-Impl-OML_0-REN_0-PO_MinKompost	241	SimC-LM_min-KP_min-Impl-OML_0-REN_0-PO_KeepHubs
172	SimC-KP_min-NW-Impl-OML_0-REN_0-PO_MinKompost	242	SimC-KP_min-NW-Impl-OML_0-REN_0-PO_KeepHubs
173	SimC-KP_min-LM_max-Impl-OML_0-REN_0-PO_MinKompost	243	SimC-KP_min-LM_max-Impl-OML_0-REN_0-PO_KeepHubs
174	SimC-KP_min-LM_min-Impl-OML_0-REN_0-PO_MinKompost	244	SimC-KP_min-LM_min-Impl-OML_0-REN_0-PO_KeepHubs
175	SimC-KP_min-KP_min-Impl-OML_0-REN_0-PO_MinKompost	245	SimC-KP_min-KP_min-Impl-OML_0-REN_0-PO_KeepHubs
176	Zickzack-OML_0-REN_0-PO_SmallestIndex	246	Zickzack-OML_0-REN_1-PO_None
177	SucC-LM_max-FIFO-KP_min-OML_0-REN_0-PO_SmallestIndex	247	SucC-LM_max-FIFO-KP_min-OML_0-REN_1-PO_None
178	SucC-LM_max-FIFO-LM_max-OML_0-REN_0-PO_SmallestIndex	248	SucC-LM_max-FIFO-LM_max-OML_0-REN_1-PO_None
179	SucC-LM_max-FIFO-NW-OML_0-REN_0-PO_SmallestIndex	249	SucC-LM_max-FIFO-NW-OML_0-REN_1-PO_None
180	SucC-LM_max-LIFO-KP_min-OML_0-REN_0-PO_SmallestIndex	250	SucC-LM_max-LIFO-KP_min-OML_0-REN_1-PO_None
181	SucC-LM_max-LIFO-LM_max-OML_0-REN_0-PO_SmallestIndex	251	SucC-LM_max-LIFO-LM_max-OML_0-REN_1-PO_None
182	SucC-LM_max-LIFO-NW-OML_0-REN_0-PO_SmallestIndex	252	SucC-LM_max-LIFO-NW-OML_0-REN_1-PO_None
183	SucC-LM_min-FIFO-KP_min-OML_0-REN_0-PO_SmallestIndex	253	SucC-LM_min-FIFO-KP_min-OML_0-REN_1-PO_None
184	SucC-LM_min-FIFO-LM_max-OML_0-REN_0-PO_SmallestIndex	254	SucC-LM_min-FIFO-LM_max-OML_0-REN_1-PO_None
185	SucC-LM_min-FIFO-NW-OML_0-REN_0-PO_SmallestIndex	255	SucC-LM_min-FIFO-NW-OML_0-REN_1-PO_None
186	SucC-LM_min-LIFO-KP_min-OML_0-REN_0-PO_SmallestIndex	256	SucC-LM_min-LIFO-KP_min-OML_0-REN_1-PO_None
187	SucC-LM_min-LIFO-LM_max-OML_0-REN_0-PO_SmallestIndex	257	SucC-LM_min-LIFO-LM_max-OML_0-REN_1-PO_None
188	SucC-LM_min-LIFO-NW-OML_0-REN_0-PO_SmallestIndex	258	SucC-LM_min-LIFO-NW-OML_0-REN_1-PO_None
189	SucC-NW-FIFO-KP_min-OML_0-REN_0-PO_SmallestIndex	259	SucC-NW-FIFO-KP_min-OML_0-REN_1-PO_None
190	SucC-NW-FIFO-LM_max-OML_0-REN_0-PO_SmallestIndex	260	SucC-NW-FIFO-LM_max-OML_0-REN_1-PO_None
191	SucC-NW-FIFO-NW-OML_0-REN_0-PO_SmallestIndex	261	SucC-NW-FIFO-NW-OML_0-REN_1-PO_None
192	SucC-NW-LIFO-KP_min-OML_0-REN_0-PO_SmallestIndex	262	SucC-NW-LIFO-KP_min-OML_0-REN_1-PO_None
193	SucC-NW-LIFO-LM_max-OML_0-REN_0-PO_SmallestIndex	263	SucC-NW-LIFO-LM_max-OML_0-REN_1-PO_None
194	SucC-NW-LIFO-NW-OML_0-REN_0-PO_SmallestIndex	264	SucC-NW-LIFO-NW-OML_0-REN_1-PO_None
195	SimC-NW-NW-Impl-OML_0-REN_0-PO_SmallestIndex	265	SimC-NW-NW-Impl-OML_0-REN_1-PO_None
196	SimC-NW-LM_max-Impl-OML_0-REN_0-PO_SmallestIndex	266	SimC-NW-LM_max-Impl-OML_0-REN_1-PO_None
197	SimC-NW-LM_min-Impl-OML_0-REN_0-PO_SmallestIndex	267	SimC-NW-LM_min-Impl-OML_0-REN_1-PO_None
198	SimC-NW-KP_min-Impl-OML_0-REN_0-PO_SmallestIndex	268	SimC-NW-KP_min-Impl-OML_0-REN_1-PO_None
199	SimC-LM_max-NW-Impl-OML_0-REN_0-PO_SmallestIndex	269	SimC-LM_max-NW-Impl-OML_0-REN_1-PO_None
200	SimC-LM_max-LM_max-Impl-OML_0-REN_0-PO_SmallestIndex	270	SimC-LM_max-LM_max-Impl-OML_0-REN_1-PO_None
201	SimC-LM_max-LM_min-Impl-OML_0-REN_0-PO_SmallestIndex	271	SimC-LM_max-LM_min-Impl-OML_0-REN_1-PO_None
202	SimC-LM_max-KP_min-Impl-OML_0-REN_0-PO_SmallestIndex	272	SimC-LM_max-KP_min-Impl-OML_0-REN_1-PO_None
203	SimC-LM_min-NW-Impl-OML_0-REN_0-PO_SmallestIndex	273	SimC-LM_min-NW-Impl-OML_0-REN_1-PO_None
204	SimC-LM_min-LM_max-Impl-OML_0-REN_0-PO_SmallestIndex	274	SimC-LM_min-LM_max-Impl-OML_0-REN_1-PO_None
205	SimC-LM_min-LM_min-Impl-OML_0-REN_0-PO_SmallestIndex	275	SimC-LM_min-LM_min-Impl-OML_0-REN_1-PO_None
206	SimC-LM_min-KP_min-Impl-OML_0-REN_0-PO_SmallestIndex	276	SimC-LM_min-KP_min-Impl-OML_0-REN_1-PO_None
207	SimC-KP_min-NW-Impl-OML_0-REN_0-PO_SmallestIndex	277	SimC-KP_min-NW-Impl-OML_0-REN_1-PO_None
208	SimC-KP_min-LM_max-Impl-OML_0-REN_0-PO_SmallestIndex	278	SimC-KP_min-LM_max-Impl-OML_0-REN_1-PO_None
209	SimC-KP_min-LM_min-Impl-OML_0-REN_0-PO_SmallestIndex	279	SimC-KP_min-LM_min-Impl-OML_0-REN_1-PO_None
210	SimC-KP_min-KP_min-Impl-OML_0-REN_0-PO_SmallestIndex	280	SimC-KP_min-KP_min-Impl-OML_0-REN_1-PO_None

ID	Verfahren	ID	Verfahren
281	Zickzack-OML_0-REN_1-PO_MaxLaub	351	Zickzack-OML_0-REN_1-PO_Median
282	SucC-LM_max-FIFO-KP_min-OML_0-REN_1-PO_MaxLaub	352	SucC-LM_max-FIFO-KP_min-OML_0-REN_1-PO_Median
283	SucC-LM_max-FIFO-LM_max-OML_0-REN_1-PO_MaxLaub	353	SucC-LM_max-FIFO-LM_max-OML_0-REN_1-PO_Median
284	SucC-LM_max-FIFO-NW-OML_0-REN_1-PO_MaxLaub	354	SucC-LM_max-FIFO-NW-OML_0-REN_1-PO_Median
285	SucC-LM_max-LIFO-KP_min-OML_0-REN_1-PO_MaxLaub	355	SucC-LM_max-LIFO-KP_min-OML_0-REN_1-PO_Median
286	SucC-LM_max-LIFO-LM_max-OML_0-REN_1-PO_MaxLaub	356	SucC-LM_max-LIFO-LM_max-OML_0-REN_1-PO_Median
287	SucC-LM_max-LIFO-NW-OML_0-REN_1-PO_MaxLaub	357	SucC-LM_max-LIFO-NW-OML_0-REN_1-PO_Median
288	SucC-LM_min-FIFO-KP_min-OML_0-REN_1-PO_MaxLaub	358	SucC-LM_min-FIFO-KP_min-OML_0-REN_1-PO_Median
289	SucC-LM_min-FIFO-LM_max-OML_0-REN_1-PO_MaxLaub	359	SucC-LM_min-FIFO-LM_max-OML_0-REN_1-PO_Median
290	SucC-LM_min-FIFO-NW-OML_0-REN_1-PO_MaxLaub	360	SucC-LM_min-FIFO-NW-OML_0-REN_1-PO_Median
291	SucC-LM_min-LIFO-KP_min-OML_0-REN_1-PO_MaxLaub	361	SucC-LM_min-LIFO-KP_min-OML_0-REN_1-PO_Median
292	SucC-LM_min-LIFO-LM_max-OML_0-REN_1-PO_MaxLaub	362	SucC-LM_min-LIFO-LM_max-OML_0-REN_1-PO_Median
293	SucC-LM_min-LIFO-NW-OML_0-REN_1-PO_MaxLaub	363	SucC-LM_min-LIFO-NW-OML_0-REN_1-PO_Median
294	SucC-NW-FIFO-KP_min-OML_0-REN_1-PO_MaxLaub	364	SucC-NW-FIFO-KP_min-OML_0-REN_1-PO_Median
295	SucC-NW-FIFO-LM_max-OML_0-REN_1-PO_MaxLaub	365	SucC-NW-FIFO-LM_max-OML_0-REN_1-PO_Median
296	SucC-NW-FIFO-NW-OML_0-REN_1-PO_MaxLaub	366	SucC-NW-FIFO-NW-OML_0-REN_1-PO_Median
297	SucC-NW-LIFO-KP_min-OML_0-REN_1-PO_MaxLaub	367	SucC-NW-LIFO-KP_min-OML_0-REN_1-PO_Median
298	SucC-NW-LIFO-LM_max-OML_0-REN_1-PO_MaxLaub	368	SucC-NW-LIFO-LM_max-OML_0-REN_1-PO_Median
299	SucC-NW-LIFO-NW-OML_0-REN_1-PO_MaxLaub	369	SucC-NW-LIFO-NW-OML_0-REN_1-PO_Median
300	SimC-NW-NW-Impl-OML_0-REN_1-PO_MaxLaub	370	SimC-NW-NW-Impl-OML_0-REN_1-PO_Median
301	SimC-NW-LM_max-Impl-OML_0-REN_1-PO_MaxLaub	371	SimC-NW-LM_max-Impl-OML_0-REN_1-PO_Median
302	SimC-NW-LM_min-Impl-OML_0-REN_1-PO_MaxLaub	372	SimC-NW-LM_min-Impl-OML_0-REN_1-PO_Median
303	SimC-NW-KP_min-Impl-OML_0-REN_1-PO_MaxLaub	373	SimC-NW-KP_min-Impl-OML_0-REN_1-PO_Median
304	SimC-LM_max-NW-Impl-OML_0-REN_1-PO_MaxLaub	374	SimC-LM_max-NW-Impl-OML_0-REN_1-PO_Median
305	SimC-LM_max-LM_max-Impl-OML_0-REN_1-PO_MaxLaub	375	SimC-LM_max-LM_max-Impl-OML_0-REN_1-PO_Median
306	SimC-LM_max-LM_min-Impl-OML_0-REN_1-PO_MaxLaub	376	SimC-LM_max-LM_min-Impl-OML_0-REN_1-PO_Median
307	SimC-LM_max-KP_min-Impl-OML_0-REN_1-PO_MaxLaub	377	SimC-LM_max-KP_min-Impl-OML_0-REN_1-PO_Median
308	SimC-LM_min-NW-Impl-OML_0-REN_1-PO_MaxLaub	378	SimC-LM_min-NW-Impl-OML_0-REN_1-PO_Median
309	SimC-LM_min-LM_max-Impl-OML_0-REN_1-PO_MaxLaub	379	SimC-LM_min-LM_max-Impl-OML_0-REN_1-PO_Median
310	SimC-LM_min-LM_min-Impl-OML_0-REN_1-PO_MaxLaub	380	SimC-LM_min-LM_min-Impl-OML_0-REN_1-PO_Median
311	SimC-LM_min-KP_min-Impl-OML_0-REN_1-PO_MaxLaub	381	SimC-LM_min-KP_min-Impl-OML_0-REN_1-PO_Median
312	SimC-KP_min-NW-Impl-OML_0-REN_1-PO_MaxLaub	382	SimC-KP_min-NW-Impl-OML_0-REN_1-PO_Median
313	SimC-KP_min-LM_max-Impl-OML_0-REN_1-PO_MaxLaub	383	SimC-KP_min-LM_max-Impl-OML_0-REN_1-PO_Median
314	SimC-KP_min-LM_min-Impl-OML_0-REN_1-PO_MaxLaub	384	SimC-KP_min-LM_min-Impl-OML_0-REN_1-PO_Median
315	SimC-KP_min-KP_min-Impl-OML_0-REN_1-PO_MaxLaub	385	SimC-KP_min-KP_min-Impl-OML_0-REN_1-PO_Median
316	Zickzack-OML_0-REN_1-PO_MinLaub	386	Zickzack-OML_0-REN_1-PO_MinKompost
317	SucC-LM_max-FIFO-KP_min-OML_0-REN_1-PO_MinLaub	387	SucC-LM_max-FIFO-KP_min-OML_0-REN_1-PO_MinKompost
318	SucC-LM_max-FIFO-LM_max-OML_0-REN_1-PO_MinLaub	388	SucC-LM_max-FIFO-LM_max-OML_0-REN_1-PO_MinKompost
319	SucC-LM_max-FIFO-NW-OML_0-REN_1-PO_MinLaub	389	SucC-LM_max-FIFO-NW-OML_0-REN_1-PO_MinKompost
320	SucC-LM_max-LIFO-KP_min-OML_0-REN_1-PO_MinLaub	390	SucC-LM_max-LIFO-KP_min-OML_0-REN_1-PO_MinKompost
321	SucC-LM_max-LIFO-LM_max-OML_0-REN_1-PO_MinLaub	391	SucC-LM_max-LIFO-LM_max-OML_0-REN_1-PO_MinKompost
322	SucC-LM_max-LIFO-NW-OML_0-REN_1-PO_MinLaub	392	SucC-LM_max-LIFO-NW-OML_0-REN_1-PO_MinKompost
323	SucC-LM_min-FIFO-KP_min-OML_0-REN_1-PO_MinLaub	393	SucC-LM_min-FIFO-KP_min-OML_0-REN_1-PO_MinKompost
324	SucC-LM_min-FIFO-LM_max-OML_0-REN_1-PO_MinLaub	394	SucC-LM_min-FIFO-LM_max-OML_0-REN_1-PO_MinKompost
325	SucC-LM_min-FIFO-NW-OML_0-REN_1-PO_MinLaub	395	SucC-LM_min-FIFO-NW-OML_0-REN_1-PO_MinKompost
326	SucC-LM_min-LIFO-KP_min-OML_0-REN_1-PO_MinLaub	396	SucC-LM_min-LIFO-KP_min-OML_0-REN_1-PO_MinKompost
327	SucC-LM_min-LIFO-LM_max-OML_0-REN_1-PO_MinLaub	397	SucC-LM_min-LIFO-LM_max-OML_0-REN_1-PO_MinKompost
328	SucC-LM_min-LIFO-NW-OML_0-REN_1-PO_MinLaub	398	SucC-LM_min-LIFO-NW-OML_0-REN_1-PO_MinKompost
329	SucC-NW-FIFO-KP_min-OML_0-REN_1-PO_MinLaub	399	SucC-NW-FIFO-KP_min-OML_0-REN_1-PO_MinKompost
330	SucC-NW-FIFO-LM_max-OML_0-REN_1-PO_MinLaub	400	SucC-NW-FIFO-LM_max-OML_0-REN_1-PO_MinKompost
331	SucC-NW-FIFO-NW-OML_0-REN_1-PO_MinLaub	401	SucC-NW-FIFO-NW-OML_0-REN_1-PO_MinKompost
332	SucC-NW-LIFO-KP_min-OML_0-REN_1-PO_MinLaub	402	SucC-NW-LIFO-KP_min-OML_0-REN_1-PO_MinKompost
333	SucC-NW-LIFO-LM_max-OML_0-REN_1-PO_MinLaub	403	SucC-NW-LIFO-LM_max-OML_0-REN_1-PO_MinKompost
334	SucC-NW-LIFO-NW-OML_0-REN_1-PO_MinLaub	404	SucC-NW-LIFO-NW-OML_0-REN_1-PO_MinKompost
335	SimC-NW-NW-Impl-OML_0-REN_1-PO_MinLaub	405	SimC-NW-NW-Impl-OML_0-REN_1-PO_MinKompost
336	SimC-NW-LM_max-Impl-OML_0-REN_1-PO_MinLaub	406	SimC-NW-LM_max-Impl-OML_0-REN_1-PO_MinKompost
337	SimC-NW-LM_min-Impl-OML_0-REN_1-PO_MinLaub	407	SimC-NW-LM_min-Impl-OML_0-REN_1-PO_MinKompost
338	SimC-NW-KP_min-Impl-OML_0-REN_1-PO_MinLaub	408	SimC-NW-KP_min-Impl-OML_0-REN_1-PO_MinKompost
339	SimC-LM_max-NW-Impl-OML_0-REN_1-PO_MinLaub	409	SimC-LM_max-NW-Impl-OML_0-REN_1-PO_MinKompost
340	SimC-LM_max-LM_max-Impl-OML_0-REN_1-PO_MinLaub	410	SimC-LM_max-LM_max-Impl-OML_0-REN_1-PO_MinKompost
341	SimC-LM_max-LM_min-Impl-OML_0-REN_1-PO_MinLaub	411	SimC-LM_max-LM_min-Impl-OML_0-REN_1-PO_MinKompost
342	SimC-LM_max-KP_min-Impl-OML_0-REN_1-PO_MinLaub	412	SimC-LM_max-KP_min-Impl-OML_0-REN_1-PO_MinKompost
343	SimC-LM_min-NW-Impl-OML_0-REN_1-PO_MinLaub	413	SimC-LM_min-NW-Impl-OML_0-REN_1-PO_MinKompost
344	SimC-LM_min-LM_max-Impl-OML_0-REN_1-PO_MinLaub	414	SimC-LM_min-LM_max-Impl-OML_0-REN_1-PO_MinKompost
345	SimC-LM_min-LM_min-Impl-OML_0-REN_1-PO_MinLaub	415	SimC-LM_min-LM_min-Impl-OML_0-REN_1-PO_MinKompost
346	SimC-LM_min-KP_min-Impl-OML_0-REN_1-PO_MinLaub	416	SimC-LM_min-KP_min-Impl-OML_0-REN_1-PO_MinKompost
347	SimC-KP_min-NW-Impl-OML_0-REN_1-PO_MinLaub	417	SimC-KP_min-NW-Impl-OML_0-REN_1-PO_MinKompost
348	SimC-KP_min-LM_max-Impl-OML_0-REN_1-PO_MinLaub	418	SimC-KP_min-LM_max-Impl-OML_0-REN_1-PO_MinKompost
349	SimC-KP_min-LM_min-Impl-OML_0-REN_1-PO_MinLaub	419	SimC-KP_min-LM_min-Impl-OML_0-REN_1-PO_MinKompost
350	SimC-KP_min-KP_min-Impl-OML_0-REN_1-PO_MinLaub	420	SimC-KP_min-KP_min-Impl-OML_0-REN_1-PO_MinKompost

ID	Verfahren	ID	Verfahren
421	Zickzack-OML_0-REN_1-PO_SmallestIndex	491	Zickzack-OML_1-REN_0-PO_None
422	SucC-LM_max-FIFO-KP_min-OML_0-REN_1-PO_SmallestIndex	492	SucC-LM_max-FIFO-KP_min-OML_1-REN_0-PO_None
423	SucC-LM_max-FIFO-LM_max-OML_0-REN_1-PO_SmallestIndex	493	SucC-LM_max-FIFO-LM_max-OML_1-REN_0-PO_None
424	SucC-LM_max-FIFO-NW-OML_0-REN_1-PO_SmallestIndex	494	SucC-LM_max-FIFO-NW-OML_1-REN_0-PO_None
425	SucC-LM_max-LIFO-KP_min-OML_0-REN_1-PO_SmallestIndex	495	SucC-LM_max-LIFO-KP_min-OML_1-REN_0-PO_None
426	SucC-LM_max-LIFO-LM_max-OML_0-REN_1-PO_SmallestIndex	496	SucC-LM_max-LIFO-LM_max-OML_1-REN_0-PO_None
427	SucC-LM_max-LIFO-NW-OML_0-REN_1-PO_SmallestIndex	497	SucC-LM_max-LIFO-NW-OML_1-REN_0-PO_None
428	SucC-LM_min-FIFO-KP_min-OML_0-REN_1-PO_SmallestIndex	498	SucC-LM_min-FIFO-KP_min-OML_1-REN_0-PO_None
429	SucC-LM_min-FIFO-LM_max-OML_0-REN_1-PO_SmallestIndex	499	SucC-LM_min-FIFO-LM_max-OML_1-REN_0-PO_None
430	SucC-LM_min-FIFO-NW-OML_0-REN_1-PO_SmallestIndex	500	SucC-LM_min-FIFO-NW-OML_1-REN_0-PO_None
431	SucC-LM_min-LIFO-KP_min-OML_0-REN_1-PO_SmallestIndex	501	SucC-LM_min-LIFO-KP_min-OML_1-REN_0-PO_None
432	SucC-LM_min-LIFO-LM_max-OML_0-REN_1-PO_SmallestIndex	502	SucC-LM_min-LIFO-LM_max-OML_1-REN_0-PO_None
433	SucC-LM_min-LIFO-NW-OML_0-REN_1-PO_SmallestIndex	503	SucC-LM_min-LIFO-NW-OML_1-REN_0-PO_None
434	SucC-NW-FIFO-KP_min-OML_0-REN_1-PO_SmallestIndex	504	SucC-NW-FIFO-KP_min-OML_1-REN_0-PO_None
435	SucC-NW-FIFO-LM_max-OML_0-REN_1-PO_SmallestIndex	505	SucC-NW-FIFO-LM_max-OML_1-REN_0-PO_None
436	SucC-NW-FIFO-NW-OML_0-REN_1-PO_SmallestIndex	506	SucC-NW-FIFO-NW-OML_1-REN_0-PO_None
437	SucC-NW-LIFO-KP_min-OML_0-REN_1-PO_SmallestIndex	507	SucC-NW-LIFO-KP_min-OML_1-REN_0-PO_None
438	SucC-NW-LIFO-LM_max-OML_0-REN_1-PO_SmallestIndex	508	SucC-NW-LIFO-LM_max-OML_1-REN_0-PO_None
439	SucC-NW-LIFO-NW-OML_0-REN_1-PO_SmallestIndex	509	SucC-NW-LIFO-NW-OML_1-REN_0-PO_None
440	SimC-NW-NW-Impl-OML_0-REN_1-PO_SmallestIndex	510	SimC-NW-NW-Impl-OML_1-REN_0-PO_None
441	SimC-NW-LM_max-Impl-OML_0-REN_1-PO_SmallestIndex	511	SimC-NW-LM_max-Impl-OML_1-REN_0-PO_None
442	SimC-NW-LM_min-Impl-OML_0-REN_1-PO_SmallestIndex	512	SimC-NW-LM_min-Impl-OML_1-REN_0-PO_None
443	SimC-NW-KP_min-Impl-OML_0-REN_1-PO_SmallestIndex	513	SimC-NW-KP_min-Impl-OML_1-REN_0-PO_None
444	SimC-LM_max-NW-Impl-OML_0-REN_1-PO_SmallestIndex	514	SimC-LM_max-NW-Impl-OML_1-REN_0-PO_None
445	SimC-LM_max-LM_max-Impl-OML_0-REN_1-PO_SmallestIndex	515	SimC-LM_max-LM_max-Impl-OML_1-REN_0-PO_None
446	SimC-LM_max-LM_min-Impl-OML_0-REN_1-PO_SmallestIndex	516	SimC-LM_max-LM_min-Impl-OML_1-REN_0-PO_None
447	SimC-LM_max-KP_min-Impl-OML_0-REN_1-PO_SmallestIndex	517	SimC-LM_max-KP_min-Impl-OML_1-REN_0-PO_None
448	SimC-LM_min-NW-Impl-OML_0-REN_1-PO_SmallestIndex	518	SimC-LM_min-NW-Impl-OML_1-REN_0-PO_None
449	SimC-LM_min-LM_max-Impl-OML_0-REN_1-PO_SmallestIndex	519	SimC-LM_min-LM_max-Impl-OML_1-REN_0-PO_None
450	SimC-LM_min-LM_min-Impl-OML_0-REN_1-PO_SmallestIndex	520	SimC-LM_min-LM_min-Impl-OML_1-REN_0-PO_None
451	SimC-LM_min-KP_min-Impl-OML_0-REN_1-PO_SmallestIndex	521	SimC-LM_min-KP_min-Impl-OML_1-REN_0-PO_None
452	SimC-KP_min-NW-Impl-OML_0-REN_1-PO_SmallestIndex	522	SimC-KP_min-NW-Impl-OML_1-REN_0-PO_None
453	SimC-KP_min-LM_max-Impl-OML_0-REN_1-PO_SmallestIndex	523	SimC-KP_min-LM_max-Impl-OML_1-REN_0-PO_None
454	SimC-KP_min-LM_min-Impl-OML_0-REN_1-PO_SmallestIndex	524	SimC-KP_min-LM_min-Impl-OML_1-REN_0-PO_None
455	SimC-KP_min-KP_min-Impl-OML_0-REN_1-PO_SmallestIndex	525	SimC-KP_min-KP_min-Impl-OML_1-REN_0-PO_None
456	Zickzack-OML_0-REN_1-PO_KeepHubs	526	Zickzack-OML_1-REN_0-PO_MaxLaub
457	SucC-LM_max-FIFO-KP_min-OML_0-REN_1-PO_KeepHubs	527	SucC-LM_max-FIFO-KP_min-OML_1-REN_0-PO_MaxLaub
458	SucC-LM_max-FIFO-LM_max-OML_0-REN_1-PO_KeepHubs	528	SucC-LM_max-FIFO-LM_max-OML_1-REN_0-PO_MaxLaub
459	SucC-LM_max-FIFO-NW-OML_0-REN_1-PO_KeepHubs	529	SucC-LM_max-FIFO-NW-OML_1-REN_0-PO_MaxLaub
460	SucC-LM_max-LIFO-KP_min-OML_0-REN_1-PO_KeepHubs	530	SucC-LM_max-LIFO-KP_min-OML_1-REN_0-PO_MaxLaub
461	SucC-LM_max-LIFO-LM_max-OML_0-REN_1-PO_KeepHubs	531	SucC-LM_max-LIFO-LM_max-OML_1-REN_0-PO_MaxLaub
462	SucC-LM_max-LIFO-NW-OML_0-REN_1-PO_KeepHubs	532	SucC-LM_max-LIFO-NW-OML_1-REN_0-PO_MaxLaub
463	SucC-LM_min-FIFO-KP_min-OML_0-REN_1-PO_KeepHubs	533	SucC-LM_min-FIFO-KP_min-OML_1-REN_0-PO_MaxLaub
464	SucC-LM_min-FIFO-LM_max-OML_0-REN_1-PO_KeepHubs	534	SucC-LM_min-FIFO-LM_max-OML_1-REN_0-PO_MaxLaub
465	SucC-LM_min-FIFO-NW-OML_0-REN_1-PO_KeepHubs	535	SucC-LM_min-FIFO-NW-OML_1-REN_0-PO_MaxLaub
466	SucC-LM_min-LIFO-KP_min-OML_0-REN_1-PO_KeepHubs	536	SucC-LM_min-LIFO-KP_min-OML_1-REN_0-PO_MaxLaub
467	SucC-LM_min-LIFO-LM_max-OML_0-REN_1-PO_KeepHubs	537	SucC-LM_min-LIFO-LM_max-OML_1-REN_0-PO_MaxLaub
468	SucC-LM_min-LIFO-NW-OML_0-REN_1-PO_KeepHubs	538	SucC-LM_min-LIFO-NW-OML_1-REN_0-PO_MaxLaub
469	SucC-NW-FIFO-KP_min-OML_0-REN_1-PO_KeepHubs	539	SucC-NW-FIFO-KP_min-OML_1-REN_0-PO_MaxLaub
470	SucC-NW-FIFO-LM_max-OML_0-REN_1-PO_KeepHubs	540	SucC-NW-FIFO-LM_max-OML_1-REN_0-PO_MaxLaub
471	SucC-NW-FIFO-NW-OML_0-REN_1-PO_KeepHubs	541	SucC-NW-FIFO-NW-OML_1-REN_0-PO_MaxLaub
472	SucC-NW-LIFO-KP_min-OML_0-REN_1-PO_KeepHubs	542	SucC-NW-LIFO-KP_min-OML_1-REN_0-PO_MaxLaub
473	SucC-NW-LIFO-LM_max-OML_0-REN_1-PO_KeepHubs	543	SucC-NW-LIFO-LM_max-OML_1-REN_0-PO_MaxLaub
474	SucC-NW-LIFO-NW-OML_0-REN_1-PO_KeepHubs	544	SucC-NW-LIFO-NW-OML_1-REN_0-PO_MaxLaub
475	SimC-NW-NW-Impl-OML_0-REN_1-PO_KeepHubs	545	SimC-NW-NW-Impl-OML_1-REN_0-PO_MaxLaub
476	SimC-NW-LM_max-Impl-OML_0-REN_1-PO_KeepHubs	546	SimC-NW-LM_max-Impl-OML_1-REN_0-PO_MaxLaub
477	SimC-NW-LM_min-Impl-OML_0-REN_1-PO_KeepHubs	547	SimC-NW-LM_min-Impl-OML_1-REN_0-PO_MaxLaub
478	SimC-NW-KP_min-Impl-OML_0-REN_1-PO_KeepHubs	548	SimC-NW-KP_min-Impl-OML_1-REN_0-PO_MaxLaub
479	SimC-LM_max-NW-Impl-OML_0-REN_1-PO_KeepHubs	549	SimC-LM_max-NW-Impl-OML_1-REN_0-PO_MaxLaub
480	SimC-LM_max-LM_max-Impl-OML_0-REN_1-PO_KeepHubs	550	SimC-LM_max-LM_max-Impl-OML_1-REN_0-PO_MaxLaub
481	SimC-LM_max-LM_min-Impl-OML_0-REN_1-PO_KeepHubs	551	SimC-LM_max-LM_min-Impl-OML_1-REN_0-PO_MaxLaub
482	SimC-LM_max-KP_min-Impl-OML_0-REN_1-PO_KeepHubs	552	SimC-LM_max-KP_min-Impl-OML_1-REN_0-PO_MaxLaub
483	SimC-LM_min-NW-Impl-OML_0-REN_1-PO_KeepHubs	553	SimC-LM_min-NW-Impl-OML_1-REN_0-PO_MaxLaub
484	SimC-LM_min-LM_max-Impl-OML_0-REN_1-PO_KeepHubs	554	SimC-LM_min-LM_max-Impl-OML_1-REN_0-PO_MaxLaub
485	SimC-LM_min-LM_min-Impl-OML_0-REN_1-PO_KeepHubs	555	SimC-LM_min-LM_min-Impl-OML_1-REN_0-PO_MaxLaub
486	SimC-LM_min-KP_min-Impl-OML_0-REN_1-PO_KeepHubs	556	SimC-LM_min-KP_min-Impl-OML_1-REN_0-PO_MaxLaub
487	SimC-KP_min-NW-Impl-OML_0-REN_1-PO_KeepHubs	557	SimC-KP_min-NW-Impl-OML_1-REN_0-PO_MaxLaub
488	SimC-KP_min-LM_max-Impl-OML_0-REN_1-PO_KeepHubs	558	SimC-KP_min-LM_max-Impl-OML_1-REN_0-PO_MaxLaub
489	SimC-KP_min-LM_min-Impl-OML_0-REN_1-PO_KeepHubs	559	SimC-KP_min-LM_min-Impl-OML_1-REN_0-PO_MaxLaub
490	SimC-KP_min-KP_min-Impl-OML_0-REN_1-PO_KeepHubs	560	SimC-KP_min-KP_min-Impl-OML_1-REN_0-PO_MaxLaub

ID	Verfahren	ID	Verfahren
561	Zickzack-OML_1-REN_0-PO_MinLaub	631	Zickzack-OML_1-REN_0-PO_MinKompost
562	SucC-LM_max-FIFO-KP_min-OML_1-REN_0-PO_MinLaub	632	SucC-LM_max-FIFO-KP_min-OML_1-REN_0-PO_MinKompost
563	SucC-LM_max-FIFO-LM_max-OML_1-REN_0-PO_MinLaub	633	SucC-LM_max-FIFO-LM_max-OML_1-REN_0-PO_MinKompost
564	SucC-LM_max-FIFO-NW-OML_1-REN_0-PO_MinLaub	634	SucC-LM_max-FIFO-NW-OML_1-REN_0-PO_MinKompost
565	SucC-LM_max-LIFO-KP_min-OML_1-REN_0-PO_MinLaub	635	SucC-LM_max-LIFO-KP_min-OML_1-REN_0-PO_MinKompost
566	SucC-LM_max-LIFO-LM_max-OML_1-REN_0-PO_MinLaub	636	SucC-LM_max-LIFO-LM_max-OML_1-REN_0-PO_MinKompost
567	SucC-LM_max-LIFO-NW-OML_1-REN_0-PO_MinLaub	637	SucC-LM_max-LIFO-NW-OML_1-REN_0-PO_MinKompost
568	SucC-LM_min-FIFO-KP_min-OML_1-REN_0-PO_MinLaub	638	SucC-LM_min-FIFO-KP_min-OML_1-REN_0-PO_MinKompost
569	SucC-LM_min-FIFO-LM_max-OML_1-REN_0-PO_MinLaub	639	SucC-LM_min-FIFO-LM_max-OML_1-REN_0-PO_MinKompost
570	SucC-LM_min-FIFO-NW-OML_1-REN_0-PO_MinLaub	640	SucC-LM_min-FIFO-NW-OML_1-REN_0-PO_MinKompost
571	SucC-LM_min-LIFO-KP_min-OML_1-REN_0-PO_MinLaub	641	SucC-LM_min-LIFO-KP_min-OML_1-REN_0-PO_MinKompost
572	SucC-LM_min-LIFO-LM_max-OML_1-REN_0-PO_MinLaub	642	SucC-LM_min-LIFO-LM_max-OML_1-REN_0-PO_MinKompost
573	SucC-LM_min-LIFO-NW-OML_1-REN_0-PO_MinLaub	643	SucC-LM_min-LIFO-NW-OML_1-REN_0-PO_MinKompost
574	SucC-NW-FIFO-KP_min-OML_1-REN_0-PO_MinLaub	644	SucC-NW-FIFO-KP_min-OML_1-REN_0-PO_MinKompost
575	SucC-NW-FIFO-LM_max-OML_1-REN_0-PO_MinLaub	645	SucC-NW-FIFO-LM_max-OML_1-REN_0-PO_MinKompost
576	SucC-NW-FIFO-NW-OML_1-REN_0-PO_MinLaub	646	SucC-NW-FIFO-NW-OML_1-REN_0-PO_MinKompost
577	SucC-NW-LIFO-KP_min-OML_1-REN_0-PO_MinLaub	647	SucC-NW-LIFO-KP_min-OML_1-REN_0-PO_MinKompost
578	SucC-NW-LIFO-LM_max-OML_1-REN_0-PO_MinLaub	648	SucC-NW-LIFO-LM_max-OML_1-REN_0-PO_MinKompost
579	SucC-NW-LIFO-NW-OML_1-REN_0-PO_MinLaub	649	SucC-NW-LIFO-NW-OML_1-REN_0-PO_MinKompost
580	SimC-NW-NW-Impl-OML_1-REN_0-PO_MinLaub	650	SimC-NW-NW-Impl-OML_1-REN_0-PO_MinKompost
581	SimC-NW-LM_max-Impl-OML_1-REN_0-PO_MinLaub	651	SimC-NW-LM_max-Impl-OML_1-REN_0-PO_MinKompost
582	SimC-NW-LM_min-Impl-OML_1-REN_0-PO_MinLaub	652	SimC-NW-LM_min-Impl-OML_1-REN_0-PO_MinKompost
583	SimC-NW-KP_min-Impl-OML_1-REN_0-PO_MinLaub	653	SimC-NW-KP_min-Impl-OML_1-REN_0-PO_MinKompost
584	SimC-LM_max-NW-Impl-OML_1-REN_0-PO_MinLaub	654	SimC-LM_max-NW-Impl-OML_1-REN_0-PO_MinKompost
585	SimC-LM_max-LM_max-Impl-OML_1-REN_0-PO_MinLaub	655	SimC-LM_max-LM_max-Impl-OML_1-REN_0-PO_MinKompost
586	SimC-LM_max-LM_min-Impl-OML_1-REN_0-PO_MinLaub	656	SimC-LM_max-LM_min-Impl-OML_1-REN_0-PO_MinKompost
587	SimC-LM_max-KP_min-Impl-OML_1-REN_0-PO_MinLaub	657	SimC-LM_max-KP_min-Impl-OML_1-REN_0-PO_MinKompost
588	SimC-LM_min-NW-Impl-OML_1-REN_0-PO_MinLaub	658	SimC-LM_min-NW-Impl-OML_1-REN_0-PO_MinKompost
589	SimC-LM_min-LM_max-Impl-OML_1-REN_0-PO_MinLaub	659	SimC-LM_min-LM_max-Impl-OML_1-REN_0-PO_MinKompost
590	SimC-LM_min-LM_min-Impl-OML_1-REN_0-PO_MinLaub	660	SimC-LM_min-LM_min-Impl-OML_1-REN_0-PO_MinKompost
591	SimC-LM_min-KP_min-Impl-OML_1-REN_0-PO_MinLaub	661	SimC-LM_min-KP_min-Impl-OML_1-REN_0-PO_MinKompost
592	SimC-KP_min-NW-Impl-OML_1-REN_0-PO_MinLaub	662	SimC-KP_min-NW-Impl-OML_1-REN_0-PO_MinKompost
593	SimC-KP_min-LM_max-Impl-OML_1-REN_0-PO_MinLaub	663	SimC-KP_min-LM_max-Impl-OML_1-REN_0-PO_MinKompost
594	SimC-KP_min-LM_min-Impl-OML_1-REN_0-PO_MinLaub	664	SimC-KP_min-LM_min-Impl-OML_1-REN_0-PO_MinKompost
595	SimC-KP_min-KP_min-Impl-OML_1-REN_0-PO_MinLaub	665	SimC-KP_min-KP_min-Impl-OML_1-REN_0-PO_MinKompost
596	Zickzack-OML_1-REN_0-PO_Median	666	Zickzack-OML_1-REN_0-PO_SmallestIndex
597	SucC-LM_max-FIFO-KP_min-OML_1-REN_0-PO_Median	667	SucC-LM_max-FIFO-KP_min-OML_1-REN_0-PO_SmallestIndex
598	SucC-LM_max-FIFO-LM_max-OML_1-REN_0-PO_Median	668	SucC-LM_max-FIFO-LM_max-OML_1-REN_0-PO_SmallestIndex
599	SucC-LM_max-FIFO-NW-OML_1-REN_0-PO_Median	669	SucC-LM_max-FIFO-NW-OML_1-REN_0-PO_SmallestIndex
600	SucC-LM_max-LIFO-KP_min-OML_1-REN_0-PO_Median	670	SucC-LM_max-LIFO-KP_min-OML_1-REN_0-PO_SmallestIndex
601	SucC-LM_max-LIFO-LM_max-OML_1-REN_0-PO_Median	671	SucC-LM_max-LIFO-LM_max-OML_1-REN_0-PO_SmallestIndex
602	SucC-LM_max-LIFO-NW-OML_1-REN_0-PO_Median	672	SucC-LM_max-LIFO-NW-OML_1-REN_0-PO_SmallestIndex
603	SucC-LM_min-FIFO-KP_min-OML_1-REN_0-PO_Median	673	SucC-LM_min-FIFO-KP_min-OML_1-REN_0-PO_SmallestIndex
604	SucC-LM_min-FIFO-LM_max-OML_1-REN_0-PO_Median	674	SucC-LM_min-FIFO-LM_max-OML_1-REN_0-PO_SmallestIndex
605	SucC-LM_min-FIFO-NW-OML_1-REN_0-PO_Median	675	SucC-LM_min-FIFO-NW-OML_1-REN_0-PO_SmallestIndex
606	SucC-LM_min-LIFO-KP_min-OML_1-REN_0-PO_Median	676	SucC-LM_min-LIFO-KP_min-OML_1-REN_0-PO_SmallestIndex
607	SucC-LM_min-LIFO-LM_max-OML_1-REN_0-PO_Median	677	SucC-LM_min-LIFO-LM_max-OML_1-REN_0-PO_SmallestIndex
608	SucC-LM_min-LIFO-NW-OML_1-REN_0-PO_Median	678	SucC-LM_min-LIFO-NW-OML_1-REN_0-PO_SmallestIndex
609	SucC-NW-FIFO-KP_min-OML_1-REN_0-PO_Median	679	SucC-NW-FIFO-KP_min-OML_1-REN_0-PO_SmallestIndex
610	SucC-NW-FIFO-LM_max-OML_1-REN_0-PO_Median	680	SucC-NW-FIFO-LM_max-OML_1-REN_0-PO_SmallestIndex
611	SucC-NW-FIFO-NW-OML_1-REN_0-PO_Median	681	SucC-NW-FIFO-NW-OML_1-REN_0-PO_SmallestIndex
612	SucC-NW-LIFO-KP_min-OML_1-REN_0-PO_Median	682	SucC-NW-LIFO-KP_min-OML_1-REN_0-PO_SmallestIndex
613	SucC-NW-LIFO-LM_max-OML_1-REN_0-PO_Median	683	SucC-NW-LIFO-LM_max-OML_1-REN_0-PO_SmallestIndex
614	SucC-NW-LIFO-NW-OML_1-REN_0-PO_Median	684	SucC-NW-LIFO-NW-OML_1-REN_0-PO_SmallestIndex
615	SimC-NW-NW-Impl-OML_1-REN_0-PO_Median	685	SimC-NW-NW-Impl-OML_1-REN_0-PO_SmallestIndex
616	SimC-NW-LM_max-Impl-OML_1-REN_0-PO_Median	686	SimC-NW-LM_max-Impl-OML_1-REN_0-PO_SmallestIndex
617	SimC-NW-LM_min-Impl-OML_1-REN_0-PO_Median	687	SimC-NW-LM_min-Impl-OML_1-REN_0-PO_SmallestIndex
618	SimC-NW-KP_min-Impl-OML_1-REN_0-PO_Median	688	SimC-NW-KP_min-Impl-OML_1-REN_0-PO_SmallestIndex
619	SimC-LM_max-NW-Impl-OML_1-REN_0-PO_Median	689	SimC-LM_max-NW-Impl-OML_1-REN_0-PO_SmallestIndex
620	SimC-LM_max-LM_max-Impl-OML_1-REN_0-PO_Median	690	SimC-LM_max-LM_max-Impl-OML_1-REN_0-PO_SmallestIndex
621	SimC-LM_max-LM_min-Impl-OML_1-REN_0-PO_Median	691	SimC-LM_max-LM_min-Impl-OML_1-REN_0-PO_SmallestIndex
622	SimC-LM_max-KP_min-Impl-OML_1-REN_0-PO_Median	692	SimC-LM_max-KP_min-Impl-OML_1-REN_0-PO_SmallestIndex
623	SimC-LM_min-NW-Impl-OML_1-REN_0-PO_Median	693	SimC-LM_min-NW-Impl-OML_1-REN_0-PO_SmallestIndex
624	SimC-LM_min-LM_max-Impl-OML_1-REN_0-PO_Median	694	SimC-LM_min-LM_max-Impl-OML_1-REN_0-PO_SmallestIndex
625	SimC-LM_min-LM_min-Impl-OML_1-REN_0-PO_Median	695	SimC-LM_min-LM_min-Impl-OML_1-REN_0-PO_SmallestIndex
626	SimC-LM_min-KP_min-Impl-OML_1-REN_0-PO_Median	696	SimC-LM_min-KP_min-Impl-OML_1-REN_0-PO_SmallestIndex
627	SimC-KP_min-NW-Impl-OML_1-REN_0-PO_Median	697	SimC-KP_min-NW-Impl-OML_1-REN_0-PO_SmallestIndex
628	SimC-KP_min-LM_max-Impl-OML_1-REN_0-PO_Median	698	SimC-KP_min-LM_max-Impl-OML_1-REN_0-PO_SmallestIndex
629	SimC-KP_min-LM_min-Impl-OML_1-REN_0-PO_Median	699	SimC-KP_min-LM_min-Impl-OML_1-REN_0-PO_SmallestIndex
630	SimC-KP_min-KP_min-Impl-OML_1-REN_0-PO_Median	700	SimC-KP_min-KP_min-Impl-OML_1-REN_0-PO_SmallestIndex

ID	Verfahren	ID	Verfahren
701	Zickzack-OML_1-REN_0-PO_KeepHubs	771	Zickzack-OML_1-REN_1-PO_MaxLaub
702	SucC-LM_max-FIFO-KP_min-OML_1-REN_0-PO_KeepHubs	772	SucC-LM_max-FIFO-KP_min-OML_1-REN_1-PO_MaxLaub
703	SucC-LM_max-FIFO-LM_max-OML_1-REN_0-PO_KeepHubs	773	SucC-LM_max-FIFO-LM_max-OML_1-REN_1-PO_MaxLaub
704	SucC-LM_max-FIFO-NW-OML_1-REN_0-PO_KeepHubs	774	SucC-LM_max-FIFO-NW-OML_1-REN_1-PO_MaxLaub
705	SucC-LM_max-LIFO-KP_min-OML_1-REN_0-PO_KeepHubs	775	SucC-LM_max-LIFO-KP_min-OML_1-REN_1-PO_MaxLaub
706	SucC-LM_max-LIFO-LM_max-OML_1-REN_0-PO_KeepHubs	776	SucC-LM_max-LIFO-LM_max-OML_1-REN_1-PO_MaxLaub
707	SucC-LM_max-LIFO-NW-OML_1-REN_0-PO_KeepHubs	777	SucC-LM_max-LIFO-NW-OML_1-REN_1-PO_MaxLaub
708	SucC-LM_min-FIFO-KP_min-OML_1-REN_0-PO_KeepHubs	778	SucC-LM_min-FIFO-KP_min-OML_1-REN_1-PO_MaxLaub
709	SucC-LM_min-FIFO-LM_max-OML_1-REN_0-PO_KeepHubs	779	SucC-LM_min-FIFO-LM_max-OML_1-REN_1-PO_MaxLaub
710	SucC-LM_min-FIFO-NW-OML_1-REN_0-PO_KeepHubs	780	SucC-LM_min-FIFO-NW-OML_1-REN_1-PO_MaxLaub
711	SucC-LM_min-LIFO-KP_min-OML_1-REN_0-PO_KeepHubs	781	SucC-LM_min-LIFO-KP_min-OML_1-REN_1-PO_MaxLaub
712	SucC-LM_min-LIFO-LM_max-OML_1-REN_0-PO_KeepHubs	782	SucC-LM_min-LIFO-LM_max-OML_1-REN_1-PO_MaxLaub
713	SucC-LM_min-LIFO-NW-OML_1-REN_0-PO_KeepHubs	783	SucC-LM_min-LIFO-NW-OML_1-REN_1-PO_MaxLaub
714	SucC-NW-FIFO-KP_min-OML_1-REN_0-PO_KeepHubs	784	SucC-NW-FIFO-KP_min-OML_1-REN_1-PO_MaxLaub
715	SucC-NW-FIFO-LM_max-OML_1-REN_0-PO_KeepHubs	785	SucC-NW-FIFO-LM_max-OML_1-REN_1-PO_MaxLaub
716	SucC-NW-FIFO-NW-OML_1-REN_0-PO_KeepHubs	786	SucC-NW-FIFO-NW-OML_1-REN_1-PO_MaxLaub
717	SucC-NW-LIFO-KP_min-OML_1-REN_0-PO_KeepHubs	787	SucC-NW-LIFO-KP_min-OML_1-REN_1-PO_MaxLaub
718	SucC-NW-LIFO-LM_max-OML_1-REN_0-PO_KeepHubs	788	SucC-NW-LIFO-LM_max-OML_1-REN_1-PO_MaxLaub
719	SucC-NW-LIFO-NW-OML_1-REN_0-PO_KeepHubs	789	SucC-NW-LIFO-NW-OML_1-REN_1-PO_MaxLaub
720	SimC-NW-NW-Impl-OML_1-REN_0-PO_KeepHubs	790	SimC-NW-NW-Impl-OML_1-REN_1-PO_MaxLaub
721	SimC-NW-LM_max-Impl-OML_1-REN_0-PO_KeepHubs	791	SimC-NW-LM_max-Impl-OML_1-REN_1-PO_MaxLaub
722	SimC-NW-LM_min-Impl-OML_1-REN_0-PO_KeepHubs	792	SimC-NW-LM_min-Impl-OML_1-REN_1-PO_MaxLaub
723	SimC-NW-KP_min-Impl-OML_1-REN_0-PO_KeepHubs	793	SimC-NW-KP_min-Impl-OML_1-REN_1-PO_MaxLaub
724	SimC-LM_max-NW-Impl-OML_1-REN_0-PO_KeepHubs	794	SimC-LM_max-NW-Impl-OML_1-REN_1-PO_MaxLaub
725	SimC-LM_max-LM_max-Impl-OML_1-REN_0-PO_KeepHubs	795	SimC-LM_max-LM_max-Impl-OML_1-REN_1-PO_MaxLaub
726	SimC-LM_max-LM_min-Impl-OML_1-REN_0-PO_KeepHubs	796	SimC-LM_max-LM_min-Impl-OML_1-REN_1-PO_MaxLaub
727	SimC-LM_max-KP_min-Impl-OML_1-REN_0-PO_KeepHubs	797	SimC-LM_max-KP_min-Impl-OML_1-REN_1-PO_MaxLaub
728	SimC-LM_min-NW-Impl-OML_1-REN_0-PO_KeepHubs	798	SimC-LM_min-NW-Impl-OML_1-REN_1-PO_MaxLaub
729	SimC-LM_min-LM_max-Impl-OML_1-REN_0-PO_KeepHubs	799	SimC-LM_min-LM_max-Impl-OML_1-REN_1-PO_MaxLaub
730	SimC-LM_min-LM_min-Impl-OML_1-REN_0-PO_KeepHubs	800	SimC-LM_min-LM_min-Impl-OML_1-REN_1-PO_MaxLaub
731	SimC-LM_min-KP_min-Impl-OML_1-REN_0-PO_KeepHubs	801	SimC-LM_min-KP_min-Impl-OML_1-REN_1-PO_MaxLaub
732	SimC-KP_min-NW-Impl-OML_1-REN_0-PO_KeepHubs	802	SimC-KP_min-NW-Impl-OML_1-REN_1-PO_MaxLaub
733	SimC-KP_min-LM_max-Impl-OML_1-REN_0-PO_KeepHubs	803	SimC-KP_min-LM_max-Impl-OML_1-REN_1-PO_MaxLaub
734	SimC-KP_min-LM_min-Impl-OML_1-REN_0-PO_KeepHubs	804	SimC-KP_min-LM_min-Impl-OML_1-REN_1-PO_MaxLaub
735	SimC-KP_min-KP_min-Impl-OML_1-REN_0-PO_KeepHubs	805	SimC-KP_min-KP_min-Impl-OML_1-REN_1-PO_MaxLaub
736	Zickzack-OML_1-REN_1-PO_None	806	Zickzack-OML_1-REN_1-PO_MinLaub
737	SucC-LM_max-FIFO-KP_min-OML_1-REN_1-PO_None	807	SucC-LM_max-FIFO-KP_min-OML_1-REN_1-PO_MinLaub
738	SucC-LM_max-FIFO-LM_max-OML_1-REN_1-PO_None	808	SucC-LM_max-FIFO-LM_max-OML_1-REN_1-PO_MinLaub
739	SucC-LM_max-FIFO-NW-OML_1-REN_1-PO_None	809	SucC-LM_max-FIFO-NW-OML_1-REN_1-PO_MinLaub
740	SucC-LM_max-LIFO-KP_min-OML_1-REN_1-PO_None	810	SucC-LM_max-LIFO-KP_min-OML_1-REN_1-PO_MinLaub
741	SucC-LM_max-LIFO-LM_max-OML_1-REN_1-PO_None	811	SucC-LM_max-LIFO-LM_max-OML_1-REN_1-PO_MinLaub
742	SucC-LM_max-LIFO-NW-OML_1-REN_1-PO_None	812	SucC-LM_max-LIFO-NW-OML_1-REN_1-PO_MinLaub
743	SucC-LM_min-FIFO-KP_min-OML_1-REN_1-PO_None	813	SucC-LM_min-FIFO-KP_min-OML_1-REN_1-PO_MinLaub
744	SucC-LM_min-FIFO-LM_max-OML_1-REN_1-PO_None	814	SucC-LM_min-FIFO-LM_max-OML_1-REN_1-PO_MinLaub
745	SucC-LM_min-FIFO-NW-OML_1-REN_1-PO_None	815	SucC-LM_min-FIFO-NW-OML_1-REN_1-PO_MinLaub
746	SucC-LM_min-LIFO-KP_min-OML_1-REN_1-PO_None	816	SucC-LM_min-LIFO-KP_min-OML_1-REN_1-PO_MinLaub
747	SucC-LM_min-LIFO-LM_max-OML_1-REN_1-PO_None	817	SucC-LM_min-LIFO-LM_max-OML_1-REN_1-PO_MinLaub
748	SucC-LM_min-LIFO-NW-OML_1-REN_1-PO_None	818	SucC-LM_min-LIFO-NW-OML_1-REN_1-PO_MinLaub
749	SucC-NW-FIFO-KP_min-OML_1-REN_1-PO_None	819	SucC-NW-FIFO-KP_min-OML_1-REN_1-PO_MinLaub
750	SucC-NW-FIFO-LM_max-OML_1-REN_1-PO_None	820	SucC-NW-FIFO-LM_max-OML_1-REN_1-PO_MinLaub
751	SucC-NW-FIFO-NW-OML_1-REN_1-PO_None	821	SucC-NW-FIFO-NW-OML_1-REN_1-PO_MinLaub
752	SucC-NW-LIFO-KP_min-OML_1-REN_1-PO_None	822	SucC-NW-LIFO-KP_min-OML_1-REN_1-PO_MinLaub
753	SucC-NW-LIFO-LM_max-OML_1-REN_1-PO_None	823	SucC-NW-LIFO-LM_max-OML_1-REN_1-PO_MinLaub
754	SucC-NW-LIFO-NW-OML_1-REN_1-PO_None	824	SucC-NW-LIFO-NW-OML_1-REN_1-PO_MinLaub
755	SimC-NW-NW-Impl-OML_1-REN_1-PO_None	825	SimC-NW-NW-Impl-OML_1-REN_1-PO_MinLaub
756	SimC-NW-LM_max-Impl-OML_1-REN_1-PO_None	826	SimC-NW-LM_max-Impl-OML_1-REN_1-PO_MinLaub
757	SimC-NW-LM_min-Impl-OML_1-REN_1-PO_None	827	SimC-NW-LM_min-Impl-OML_1-REN_1-PO_MinLaub
758	SimC-NW-KP_min-Impl-OML_1-REN_1-PO_None	828	SimC-NW-KP_min-Impl-OML_1-REN_1-PO_MinLaub
759	SimC-LM_max-NW-Impl-OML_1-REN_1-PO_None	829	SimC-LM_max-NW-Impl-OML_1-REN_1-PO_MinLaub
760	SimC-LM_max-LM_max-Impl-OML_1-REN_1-PO_None	830	SimC-LM_max-LM_max-Impl-OML_1-REN_1-PO_MinLaub
761	SimC-LM_max-LM_min-Impl-OML_1-REN_1-PO_None	831	SimC-LM_max-LM_min-Impl-OML_1-REN_1-PO_MinLaub
762	SimC-LM_max-KP_min-Impl-OML_1-REN_1-PO_None	832	SimC-LM_max-KP_min-Impl-OML_1-REN_1-PO_MinLaub
763	SimC-LM_min-NW-Impl-OML_1-REN_1-PO_None	833	SimC-LM_min-NW-Impl-OML_1-REN_1-PO_MinLaub
764	SimC-LM_min-LM_max-Impl-OML_1-REN_1-PO_None	834	SimC-LM_min-LM_max-Impl-OML_1-REN_1-PO_MinLaub
765	SimC-LM_min-LM_min-Impl-OML_1-REN_1-PO_None	835	SimC-LM_min-LM_min-Impl-OML_1-REN_1-PO_MinLaub
766	SimC-LM_min-KP_min-Impl-OML_1-REN_1-PO_None	836	SimC-LM_min-KP_min-Impl-OML_1-REN_1-PO_MinLaub
767	SimC-KP_min-NW-Impl-OML_1-REN_1-PO_None	837	SimC-KP_min-NW-Impl-OML_1-REN_1-PO_MinLaub
768	SimC-KP_min-LM_max-Impl-OML_1-REN_1-PO_None	838	SimC-KP_min-LM_max-Impl-OML_1-REN_1-PO_MinLaub
769	SimC-KP_min-LM_min-Impl-OML_1-REN_1-PO_None	839	SimC-KP_min-LM_min-Impl-OML_1-REN_1-PO_MinLaub
770	SimC-KP_min-KP_min-Impl-OML_1-REN_1-PO_None	840	SimC-KP_min-KP_min-Impl-OML_1-REN_1-PO_MinLaub

ID	Verfahren	ID	Verfahren
841	Zickzack-OML_1-REN_1-PO_Median	911	Zickzack-OML_1-REN_1-PO_SmallestIndex
842	SucC-LM_max-FIFO-KP_min-OML_1-REN_1-PO_Median	912	SucC-LM_max-FIFO-KP_min-OML_1-REN_1-PO_SmallestIndex
843	SucC-LM_max-FIFO-LM_max-OML_1-REN_1-PO_Median	913	SucC-LM_max-FIFO-LM_max-OML_1-REN_1-PO_SmallestIndex
844	SucC-LM_max-FIFO-NW-OML_1-REN_1-PO_Median	914	SucC-LM_max-FIFO-NW-OML_1-REN_1-PO_SmallestIndex
845	SucC-LM_max-LIFO-KP_min-OML_1-REN_1-PO_Median	915	SucC-LM_max-LIFO-KP_min-OML_1-REN_1-PO_SmallestIndex
846	SucC-LM_max-LIFO-LM_max-OML_1-REN_1-PO_Median	916	SucC-LM_max-LIFO-LM_max-OML_1-REN_1-PO_SmallestIndex
847	SucC-LM_max-LIFO-NW-OML_1-REN_1-PO_Median	917	SucC-LM_max-LIFO-NW-OML_1-REN_1-PO_SmallestIndex
848	SucC-LM_min-FIFO-KP_min-OML_1-REN_1-PO_Median	918	SucC-LM_min-FIFO-KP_min-OML_1-REN_1-PO_SmallestIndex
849	SucC-LM_min-FIFO-LM_max-OML_1-REN_1-PO_Median	919	SucC-LM_min-FIFO-LM_max-OML_1-REN_1-PO_SmallestIndex
850	SucC-LM_min-FIFO-NW-OML_1-REN_1-PO_Median	920	SucC-LM_min-FIFO-NW-OML_1-REN_1-PO_SmallestIndex
851	SucC-LM_min-LIFO-KP_min-OML_1-REN_1-PO_Median	921	SucC-LM_min-LIFO-KP_min-OML_1-REN_1-PO_SmallestIndex
852	SucC-LM_min-LIFO-LM_max-OML_1-REN_1-PO_Median	922	SucC-LM_min-LIFO-LM_max-OML_1-REN_1-PO_SmallestIndex
853	SucC-LM_min-LIFO-NW-OML_1-REN_1-PO_Median	923	SucC-LM_min-LIFO-NW-OML_1-REN_1-PO_SmallestIndex
854	SucC-NW-FIFO-KP_min-OML_1-REN_1-PO_Median	924	SucC-NW-FIFO-KP_min-OML_1-REN_1-PO_SmallestIndex
855	SucC-NW-FIFO-LM_max-OML_1-REN_1-PO_Median	925	SucC-NW-FIFO-LM_max-OML_1-REN_1-PO_SmallestIndex
856	SucC-NW-FIFO-NW-OML_1-REN_1-PO_Median	926	SucC-NW-FIFO-NW-OML_1-REN_1-PO_SmallestIndex
857	SucC-NW-LIFO-KP_min-OML_1-REN_1-PO_Median	927	SucC-NW-LIFO-KP_min-OML_1-REN_1-PO_SmallestIndex
858	SucC-NW-LIFO-LM_max-OML_1-REN_1-PO_Median	928	SucC-NW-LIFO-LM_max-OML_1-REN_1-PO_SmallestIndex
859	SucC-NW-LIFO-NW-OML_1-REN_1-PO_Median	929	SucC-NW-LIFO-NW-OML_1-REN_1-PO_SmallestIndex
860	SimC-NW-NW-Impl-OML_1-REN_1-PO_Median	930	SimC-NW-NW-Impl-OML_1-REN_1-PO_SmallestIndex
861	SimC-NW-LM_max-Impl-OML_1-REN_1-PO_Median	931	SimC-NW-LM_max-Impl-OML_1-REN_1-PO_SmallestIndex
862	SimC-NW-LM_min-Impl-OML_1-REN_1-PO_Median	932	SimC-NW-LM_min-Impl-OML_1-REN_1-PO_SmallestIndex
863	SimC-NW-KP_min-Impl-OML_1-REN_1-PO_Median	933	SimC-NW-KP_min-Impl-OML_1-REN_1-PO_SmallestIndex
864	SimC-LM_max-NW-Impl-OML_1-REN_1-PO_Median	934	SimC-LM_max-NW-Impl-OML_1-REN_1-PO_SmallestIndex
865	SimC-LM_max-LM_max-Impl-OML_1-REN_1-PO_Median	935	SimC-LM_max-LM_max-Impl-OML_1-REN_1-PO_SmallestIndex
866	SimC-LM_max-LM_min-Impl-OML_1-REN_1-PO_Median	936	SimC-LM_max-LM_min-Impl-OML_1-REN_1-PO_SmallestIndex
867	SimC-LM_max-KP_min-Impl-OML_1-REN_1-PO_Median	937	SimC-LM_max-KP_min-Impl-OML_1-REN_1-PO_SmallestIndex
868	SimC-LM_min-NW-Impl-OML_1-REN_1-PO_Median	938	SimC-LM_min-NW-Impl-OML_1-REN_1-PO_SmallestIndex
869	SimC-LM_min-LM_max-Impl-OML_1-REN_1-PO_Median	939	SimC-LM_min-LM_max-Impl-OML_1-REN_1-PO_SmallestIndex
870	SimC-LM_min-LM_min-Impl-OML_1-REN_1-PO_Median	940	SimC-LM_min-LM_min-Impl-OML_1-REN_1-PO_SmallestIndex
871	SimC-LM_min-KP_min-Impl-OML_1-REN_1-PO_Median	941	SimC-LM_min-KP_min-Impl-OML_1-REN_1-PO_SmallestIndex
872	SimC-KP_min-NW-Impl-OML_1-REN_1-PO_Median	942	SimC-KP_min-NW-Impl-OML_1-REN_1-PO_SmallestIndex
873	SimC-KP_min-LM_max-Impl-OML_1-REN_1-PO_Median	943	SimC-KP_min-LM_max-Impl-OML_1-REN_1-PO_SmallestIndex
874	SimC-KP_min-LM_min-Impl-OML_1-REN_1-PO_Median	944	SimC-KP_min-LM_min-Impl-OML_1-REN_1-PO_SmallestIndex
875	SimC-KP_min-KP_min-Impl-OML_1-REN_1-PO_Median	945	SimC-KP_min-KP_min-Impl-OML_1-REN_1-PO_SmallestIndex
876	Zickzack-OML_1-REN_1-PO_MinKompost	946	Zickzack-OML_1-REN_1-PO_KeepHubs
877	SucC-LM_max-FIFO-KP_min-OML_1-REN_1-PO_MinKompost	947	SucC-LM_max-FIFO-KP_min-OML_1-REN_1-PO_KeepHubs
878	SucC-LM_max-FIFO-LM_max-OML_1-REN_1-PO_MinKompost	948	SucC-LM_max-FIFO-LM_max-OML_1-REN_1-PO_KeepHubs
879	SucC-LM_max-FIFO-NW-OML_1-REN_1-PO_MinKompost	949	SucC-LM_max-FIFO-NW-OML_1-REN_1-PO_KeepHubs
880	SucC-LM_max-LIFO-KP_min-OML_1-REN_1-PO_MinKompost	950	SucC-LM_max-LIFO-KP_min-OML_1-REN_1-PO_KeepHubs
881	SucC-LM_max-LIFO-LM_max-OML_1-REN_1-PO_MinKompost	951	SucC-LM_max-LIFO-LM_max-OML_1-REN_1-PO_KeepHubs
882	SucC-LM_max-LIFO-NW-OML_1-REN_1-PO_MinKompost	952	SucC-LM_max-LIFO-NW-OML_1-REN_1-PO_KeepHubs
883	SucC-LM_min-FIFO-KP_min-OML_1-REN_1-PO_MinKompost	953	SucC-LM_min-FIFO-KP_min-OML_1-REN_1-PO_KeepHubs
884	SucC-LM_min-FIFO-LM_max-OML_1-REN_1-PO_MinKompost	954	SucC-LM_min-FIFO-LM_max-OML_1-REN_1-PO_KeepHubs
885	SucC-LM_min-FIFO-NW-OML_1-REN_1-PO_MinKompost	955	SucC-LM_min-FIFO-NW-OML_1-REN_1-PO_KeepHubs
886	SucC-LM_min-LIFO-KP_min-OML_1-REN_1-PO_MinKompost	956	SucC-LM_min-LIFO-KP_min-OML_1-REN_1-PO_KeepHubs
887	SucC-LM_min-LIFO-LM_max-OML_1-REN_1-PO_MinKompost	957	SucC-LM_min-LIFO-LM_max-OML_1-REN_1-PO_KeepHubs
888	SucC-LM_min-LIFO-NW-OML_1-REN_1-PO_MinKompost	958	SucC-LM_min-LIFO-NW-OML_1-REN_1-PO_KeepHubs
889	SucC-NW-FIFO-KP_min-OML_1-REN_1-PO_MinKompost	959	SucC-NW-FIFO-KP_min-OML_1-REN_1-PO_KeepHubs
890	SucC-NW-FIFO-LM_max-OML_1-REN_1-PO_MinKompost	960	SucC-NW-FIFO-LM_max-OML_1-REN_1-PO_KeepHubs
891	SucC-NW-FIFO-NW-OML_1-REN_1-PO_MinKompost	961	SucC-NW-FIFO-NW-OML_1-REN_1-PO_KeepHubs
892	SucC-NW-LIFO-KP_min-OML_1-REN_1-PO_MinKompost	962	SucC-NW-LIFO-KP_min-OML_1-REN_1-PO_KeepHubs
893	SucC-NW-LIFO-LM_max-OML_1-REN_1-PO_MinKompost	963	SucC-NW-LIFO-LM_max-OML_1-REN_1-PO_KeepHubs
894	SucC-NW-LIFO-NW-OML_1-REN_1-PO_MinKompost	964	SucC-NW-LIFO-NW-OML_1-REN_1-PO_KeepHubs
895	SimC-NW-NW-Impl-OML_1-REN_1-PO_MinKompost	965	SimC-NW-NW-Impl-OML_1-REN_1-PO_KeepHubs
896	SimC-NW-LM_max-Impl-OML_1-REN_1-PO_MinKompost	966	SimC-NW-LM_max-Impl-OML_1-REN_1-PO_KeepHubs
897	SimC-NW-LM_min-Impl-OML_1-REN_1-PO_MinKompost	967	SimC-NW-LM_min-Impl-OML_1-REN_1-PO_KeepHubs
898	SimC-NW-KP_min-Impl-OML_1-REN_1-PO_MinKompost	968	SimC-NW-KP_min-Impl-OML_1-REN_1-PO_KeepHubs
899	SimC-LM_max-NW-Impl-OML_1-REN_1-PO_MinKompost	969	SimC-LM_max-NW-Impl-OML_1-REN_1-PO_KeepHubs
900	SimC-LM_max-LM_max-Impl-OML_1-REN_1-PO_MinKompost	970	SimC-LM_max-LM_max-Impl-OML_1-REN_1-PO_KeepHubs
901	SimC-LM_max-LM_min-Impl-OML_1-REN_1-PO_MinKompost	971	SimC-LM_max-LM_min-Impl-OML_1-REN_1-PO_KeepHubs
902	SimC-LM_max-KP_min-Impl-OML_1-REN_1-PO_MinKompost	972	SimC-LM_max-KP_min-Impl-OML_1-REN_1-PO_KeepHubs
903	SimC-LM_min-NW-Impl-OML_1-REN_1-PO_MinKompost	973	SimC-LM_min-NW-Impl-OML_1-REN_1-PO_KeepHubs
904	SimC-LM_min-LM_max-Impl-OML_1-REN_1-PO_MinKompost	974	SimC-LM_min-LM_max-Impl-OML_1-REN_1-PO_KeepHubs
905	SimC-LM_min-LM_min-Impl-OML_1-REN_1-PO_MinKompost	975	SimC-LM_min-LM_min-Impl-OML_1-REN_1-PO_KeepHubs
906	SimC-LM_min-KP_min-Impl-OML_1-REN_1-PO_MinKompost	976	SimC-LM_min-KP_min-Impl-OML_1-REN_1-PO_KeepHubs
907	SimC-KP_min-NW-Impl-OML_1-REN_1-PO_MinKompost	977	SimC-KP_min-NW-Impl-OML_1-REN_1-PO_KeepHubs
908	SimC-KP_min-LM_max-Impl-OML_1-REN_1-PO_MinKompost	978	SimC-KP_min-LM_max-Impl-OML_1-REN_1-PO_KeepHubs
909	SimC-KP_min-LM_min-Impl-OML_1-REN_1-PO_MinKompost	979	SimC-KP_min-LM_min-Impl-OML_1-REN_1-PO_KeepHubs
910	SimC-KP_min-KP_min-Impl-OML_1-REN_1-PO_MinKompost	980	SimC-KP_min-KP_min-Impl-OML_1-REN_1-PO_KeepHubs

Tabelle 36: *Liste der Kürzel in den Namen der deterministischen Verfahrensvarianten im MATLAB-Programm* `lhp`

Kürzel	Bedeutung
SucC	Sukzessive Clusterbildung; vgl. Abschnitt 3.1
SimC	Simultane Clusterbildung; vgl. Abschnitt 3.2
NW	Auswahl gemäß Nordwestecken-Regel bzw. Kleinster-Index-Prinzip
LM_max	Auswahl nach maximaler Laubmenge
LM_min	Auswahl nach minimaler Laubmenge
KP_min	Auswahl nach minimaler Entfernung zum Kompostfeld
FIFO	Breitensuche via Schlangenliste
LIFO	Tiefensuche via Stapelliste
OML_0	Ohne Zusatzmodul *Laubmengenobergrenzenvariation*
OML_1	Mit Zusatzmodul *Laubmengenobergrenzenvariation*
REN_0	Ohne Zusatzmodul *Aussparung laubleerer Felder*
REN_1	Mit Zusatzmodul *Aussparung laubleerer Felder*
PO_None	Keine explizite Hubbestimmung
PO_KeepHubs	Hubausrichtung unter Beibehaltung der implizit ermittelten Hubs; vgl. Abschnitt 3.3
PO_MaxLaub	Explizite Hubbestimmung nach maximaler Laubmenge
PO_MinLaub	Explizite Hubbestimmung nach minimaler Laubmenge
PO_Median	Explizite Hubbestimmung nach Median
PO_MinKompost	Explizite Hubbestimmung nach minimaler Kompostentfernung
PO_SmallestIndex	Explizite Hubbestimmung nach Nordwestecken-Regel (NW) bzw. Kleinster-Index-Prinzip

A.4 Glossar

Bezeichner	Bedeutung	Seite
$a \in \mathbb{N}$	**Feld** eines Gartens	7
$G = (V, E)$	**Graph**, durch den der Garten modellhaft beschrieben wird	10
V	**Knotenmenge** des Graphen G; entspricht den Feldern des durch G modellierten Gartens	10
E	**Kantenmenge** des Graphen G; beschreibt die Nachbarschaft der Felder des durch G modellierten Gartens	10
C	**Bewertungsmatrix** eines kantenbewerteten Graphen	11
D	**Entfernungsmatrix** eines kantenbewerteten Graphen	11
$s : V \to V$	**Nachfolgerfunktion**, wenn die *Nachbarschaftsbedingung* (2.9) gilt und der durch $s \to$*induzierte Teilgraph* $\vec{G}_s$ zyklenfrei ist	18
$\vec{G}_s = \langle V, \vec{E}_s \rangle$	durch die $\to$*Nachfolgerfunktion* s **induzierter Teilgraph** von $G = (V, E)$ mit $\vec{E}_s = \{\langle a, s(a)\rangle \mid a \in V \wedge a \neq s(a)\}$	18
$h \in V$	**Haufenfeld** oder **Hub** $\equiv$ Senke des durch die $\to$*Nachfolgerfunktion* s $\to$*induzierten Teilgraphen* $\vec{G}_s$, d.h. $s(h) = h$	17
$Hub(s)$	**Hubmenge** $\equiv$ Menge aller $\to$*Haufenfelder* ($\to$*Hubs*), welche durch die $\to$*Nachfolgerfunktion* s induziert werden	19
$q \in V$	**Harkquelle** $\equiv$ Quelle des durch die $\to$*Nachfolgerfunktion* s $\to$*induzierten Teilgraphen* $\vec{G}_s$	18
$Q(s)$	**Harkquellenmenge** $\equiv$ Menge aller $\to$*Harkquellen*, welche durch die $\to$*Nachfolgerfunktion* s induziert werden	18
$K \in V$	**Kompostfeld** $\equiv$ Kompoststelle des Gartens	25
$M(a)$	initiale **Laubmenge** in $\to$*Feld* a; entspricht der Knotenbewertung des $\to$*Graphen* G	8
$M^*(a)$	**aktuelle Laubmenge** in $\to$*Feld* a; gibt die (zwischenzeitlich) kumulierte Laubmenge in Feld a zum Zeitpunkt des Harkens von Feld a zu einem Nachbarfeld an	12
M_k	**kritische Laubharkmenge**; entspricht der Laubmenge, die noch mit einem einzigen Harkzug bewegt werden kann	13
$\overline{M}, \overline{M}(a)$	**maximale Laubmenge** (von $\to$*Feld* a); entspricht der Laubmenge, die maximal im Feld a angehäuft werden kann	14
$\widetilde{M}$	**empfohlene Laubmengenobergrenze**; entspricht der Laubmenge, die im Hinblick auf die Lösungsgüte der intuitiven Heuristiken maximal pro Feld angehäuft werden sollte	40
$M_s^* : V \to \mathbb{R}_{\geq}$	die durch die $\to$*Nachfolgerfunktion* s induzierte **Anhäufungsfunktion** gemäß Bildungsvorschrift (2.10)	19,27

Bezeichner	Bedeutung	Seite
α_H, $\alpha_H(a,b)$	**Harkaufwandsfaktor** oder **Harkparameter**	12
$HA(a,b)$	einzelner **Harkaufwand**; entspricht dem Aufwand für das Harken der aktuellen Laubmenge von Feld a in ein benachbartes Feld b	12f
$HA_\Sigma(s)$	**Harkaufwandsumme**; entspricht dem Gesamtaufwand des produktiven Harkprozesses, der durch die →*Nachfolgerfunktion* s beschrieben wird	19
α_W, $\alpha_W(a,q)$	**Wegeaufwandsfaktor** oder **Wegeparameter**	23
$W(s)$	**unproduktive Weglänge**; entspricht der Gesamtlänge aller unproduktiven Wege, welche durch die →*Nachfolgerfunktion* s hervorgerufen werden	23
$WA_\Sigma(s)$	**unproduktiver Wegeaufwand**; entspricht dem Gesamtaufwand für unproduktive Wege des durch die →*Nachfolgerfunktion* s beschriebenen Harkprozesses	23f
α_T	**Transportaufwandsfaktor** oder **Transportparameter**	25
$\overline{T}$	**maximales Ladevolumen** des Transportmittels	26
γ	**variabler Ladeaufwand** ≡ variabler Aufwandsanteil für Auf- und Abladen einer Laubmenge	25
σ	**fixer Ladeaufwand** ≡ fixer Aufwandsanteil für Auf- und Abladen einer Laubmenge	25
$TA(h)$	**Transportaufwand** von →*Hub* h zum →*Kompostfeld* K; entspricht dem Aufwand für das Transportieren der kumulierten Laubmenge von →*Hub* h zum →*Kompostfeld* K, einschließlich der Aufwände für das Auf- und Abladen	25
$TA_\Sigma(s)$	**Transportaufwandsumme**; ergibt sich als gesamter Transportaufwand aller Touren zwischen allen →*Hubs* bzgl. →*Nachfolgerfunktion* s und →*Kompostfeld*, einschließlich aller Aufwände für das Auf- und Abladen	26
$M_s^{**}(h)$	eine nach entsprechend vielen Pendeltouren verbleibende **Laubrestmenge** beim →*Hub* h	26
$\pi : V \leftrightarrow V$	Permutation zur **Ausrichtung** einer →*Nachfolgerfunktion* s	23
$R(a)$	**Erreichbarkeitsmenge**; enthält alle Knoten $i \in V$, von denen aus der Knoten a entlang einer Pfeilfolge im →*induzierten Teilgraph* $\vec{G}_s$ erreichbar ist	19,27
$C(h)$	**Cluster**; entspricht der →*Erreichbarkeitsmenge* einer Senke h des →*induzierten Teilgraphen* $\vec{G}_s$	27
$\Lambda(G)$	**Lösungsraum** des Laubharkproblems	27
$K(s)$	**Kostenfunktion** als Zielfunktion des Laubharkproblems	27

Bezeichner	Bedeutung	Seite
$F(s)$	**Fitnessfunktion**; beschreibt die Fitness eines Chromosomes	48
P^i	**Population** zum Iterationsschritt i	48
$\tilde{P}^i$	**Zwischenpopulation** zum Iterationsschritt i	48
s_k^i	k-te **Chromosom** der Population P^i zum Iterationsschritt i	48
$\tilde{s}_k^i$	k-te **Chromosom** der Zwischenpopulation $\tilde{P}^i$ zum Iterationsschritt i	48
$\overline{F^i}$	**Gesamt-Fitness** einer Population P^i	48
$N(s_k^i), N_k^i$	**Normierte Fitnessfunktion** des Chromosom s_k^i	48
$\hat{N}(s_k^i), \hat{N}_k^i$	**Akkumulierte normierte Fitnessfunktion** des Chromosom s_k^i	48
$W_{R1}(s), W_{RM}(s)$	**Überlebenswahrscheinlichkeit** eines Chromosomes s	49
$H_{RM}^{max}(s), H_{RM}^{min}(s)$	maximale und minimale **Häufigkeit** eines Chromosomes	50
ns	Anzahl der **Scout-Bienen** beim Bienenalgorithmus	53
nb, ne	Anzahl der **Best-Bienen** und **Elite-Bienen** zu den guten und sehr guten Futterstellen	53
nrb, nre	Anzahl der durch die Best-Bienen und Elite-Bienen **rekrutierten Bienen**	53

Die Autoren

Karim Abdelhak (M.Sc.)
Hochschule Bielefeld (HSBI)
University of Applied Sciences and Arts
Fachbereich Ingenieurwissenschaften und Mathematik
karim.abdelhak@hsbi.de

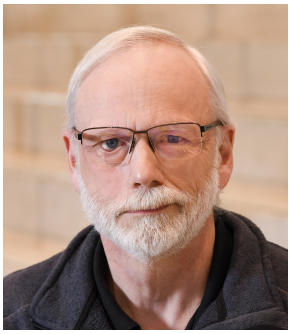
Fachgebiete: Graphentheorie, Optimierung, Symbolische Behandlung großer hybrider differential-algebraischer Gleichungssysteme

Seit dem Abschluss als Master of Science in Optimierung & Simulation 2018 als Wissenschaftlicher Mitarbeiter an der Fachhochschule Bielefeld (jüngst umbenannt in Hochschule Bielefeld – University of Applied Sciences and Arts (HSBI)) im Auftrag des *Open Source Modelica Consortium* (OSMC) tätig. Seit 2019 Promotionsstudent für Informatik an der Technischen Fakultät der Universität Bielefeld. Mitglied des Forschungsschwerpunktes *Angewandte Mathematische Modellierung & Optimierung* (FSP AMMO) der Hochschule Bielefeld.

Prof. Dr. phil. Dipl. -Math. Bernhard Bachmann
Hochschule Bielefeld (HSBI)
University of Applied Sciences and Arts
Fachbereich Ingenieurwissenschaften und Mathematik
bernhard.bachmann@hsbi.de

Fachgebiete: Numerische Mathematik, Optimierung, Symbolische und numerische Behandlung großer hybrider differential-algebraischer Gleichungssysteme

Seit 1999 als Professor für Mathematik und ihre technischen Anwendungen an der Fachhochschule Bielefeld (jüngst umbenannt in Hochschule Bielefeld – University of Applied Sciences and Arts (HSBI)) tätig und lehrt dort im Bachelor-Studiengang *Angewandte Mathematik* und im Master-Studiengang *Optimierung & Simulation* des Fachbereichs *Ingenieurwissenschaften & Mathematik*. Gründungsmitglied des Forschungsschwerpunktes *Angewandte Mathematische Modellierung & Optimierung* (FSP AMMO) der Hochschule Bielefeld. Gründungsmitglied der *Modelica Association*. Gründungs- und Vorstandsmitglied des *Open Source Modelica Consortium* (OSMC). Nationale und internationale F&E-Projekte im Bereich der Modellierung, Simulation und Optimierung hybrider dynamischer Systeme.



Dipl.-Wirt.Math Ralf Derdau
Hochschule Bielefeld (HSBI)
University of Applied Sciences and Arts
Fachbereich Ingenieurwissenschaften und Mathematik
ralf.derdau@hsbi.de

Fachgebiete: Wirtschaftsmathematik, Informatik

Seit 1994 als wissenschaftlicher Mitarbeiter an der Fachhochschule Bielefeld (jüngst umbenannt in Hochschule Bielefeld – University of Applied Sciences and Arts (HSBI)) und dort im Bachelor-Studiengang *Angewandte Mathematik* und im Master-Studiengang *Optimierung & Simulation* des Fachbereichs *Ingenieurwissenschaften & Mathematik* tätig. Aufgabengebiete sind die Bereitstellung und der Support der informationstechnischen Infrastruktur des Studienganges Angewandte Mathematik, sowie die Schulung dort eingesetzter Mathematik- und Simulationssoftware.

Andreas Hartmann (M.Sc.)
engineering people Stuttgart GmbH
hartan@7x.de

Fachgebiete: Softwareentwicklung, MATLAB

Linux-Enthusiast und begeisterter Open-Source-Entwickler, gegenwärtig tätig als Embedded-Software-Entwickler bei der engineering people Stuttgart GmbH. Studierte an der Fachhochschule Bielefeld (jüngst umbenannt in Hochschule Bielefeld – University of Applied Sciences and Arts (HSBI)) von 2016 bis 2020 im Bachelor-Studiengang *Mechatronik*, anschließend bis 2023 im Master-Studiengang *BioMechatronik*. Sammelte in dieser Zeit viele Erfahrungen mit MATLAB bei der Bearbeitung von Projekten zu den Themen Embedded-Entwicklung auf ARM Cortex-M, Optimierung und Simulation (mit und ohne Simulink) sowie digitale Signalverarbeitung mit Schwerpunkt auf Mehrkanal-Audiosignale.

Prof. Dr. rer. pol. Dipl. -Math. Hermann-Josef Kruse
Hochschule Bielefeld (HSBI)
University of Applied Sciences and Arts
Fachbereich Ingenieurwissenschaften und Mathematik
hj.kruse@freenet.de

Fachgebiete: Wirtschaftsmathematik, Operations Research

Von 1995 bis 2021 als Professor für Wirtschaftsmathematik (insbesondere Operations Research) an der Fachhochschule Bielefeld (jüngst umbenannt in Hochschule Bielefeld – University of Applied Sciences and Arts (HSBI)) tätig und lehrte dort im Bachelor-Studiengang *Angewandte Mathematik* und im Master-Studiengang *Optimierung & Simulation* des Fachbereichs *Ingenieurwissenschaften & Mathematik* (emeritiert seit 2019). Gründungsmitglied des Forschungsschwerpunktes *Angewandte Mathematische Modellierung & Optimierung* (FSP AMMO) der Hochschule Bielefeld. F&E-Projekte im Bereich der Optimierung und Simulation diskreter Systeme zur Entscheidungsunterstützung bei betrieblichen Problemstellungen.

Bisher in der Forschungsreihe des Fachbereichs Ingenieurwissenschaften und Mathematik erschienen:

- Band 1:
  Hybrid Modeling and Optimization of Biological Processes
  Sabrina Proß (Juli 2013)

- Band 2:
  Petri-Netz-Formalismen und Lösungsansätze für allgemeine Konfliktsituationen bei Feuerprozessen in Petri-Netz-Modellen
  Bernhard Bachmann, Timo Kleine-Döpke, Hermann-Josef Kruse, Lennart Ochel, Sabrina Proß (November 2014)

- Band 3:
  Application Techniques of Endophytes
  Desiree Jakobs-Schönwandt, Matthias Döring, Anant Patel (November 2014)

- Band 4:
  Angewandte mathematische Modellierung und Optimierung
  Ausgewählte Modelle, Methoden, Fallstudien
  Hermann-Josef Kruse (Hrsg.), Timo Lask (Hrsg.) (Februar 2016)

Forschungsreihe des Fachbereichs
Ingenieurwissenschaften und Mathematik

Herausgeber
Hochschule Bielefeld (HSBI)
Fachbereich Ingenieurwissenschaften und Mathematik

Band 5
Juli 2023

ISSN: 2196-6192
DOI: 10.57720/3321