Carl von Ossietzky Universität Oldenburg

Informatik
Masterarbeit

Entwicklung von prädiktiven und reaktiven Lösungsansätzen für das MRCPSP auf Basis von Metaheuristiken

---

vorgelegt von:
Autor: René Kuchenbuch, B. Sc.
Studiengang: Informatik (M. Sc.)
Matr. Nr.: 5896997

Betreuender Gutachter:
apl. Prof. Dr.-Ing. Jürgen Sauer

Zweiter Gutachter:
Julius Möller, M. Sc.

Abteilung Systemanalyse und -optimierung
Department für Informatik

Oldenburg, 9. März 2022

Abstract (deutsch)

Das Multi-Mode Resource-Constrained Project Scheduling Problem (MRCPSP) stellt ein NP-vollständiges Optimierungsproblem dar, welches sich mit der Erzeugung von Projektplänen mit Ressourcenbeschränkungen befasst. Unsicherheiten, wie z. B. Verspätungen führen dazu, dass ein Projektplan nicht mehr in geplanter Zeit vollendet werden kann.

Das Ziel der Masterarbeit befasst sich zunächst mit der Suche von Basiszeitplänen für Projekte mit Hilfe von Metaheuristiken, welche die Projektdauer minimieren sollen. Anschließend gilt es prädiktive und reaktive Verfahren zu selektieren um die einhergehenden Verspätungen durch Unsicherheiten zu minimieren. „Welche Auswirkungen haben prädiktive und reaktive Ansätze für das MRCPSP auf Basis von metaheuristischen Algorithmen bei der Erstellung von Zeitplänen in Bezug auf die Projektdauer bei Verspätungen?“ stellt die Forschungsfrage der Thesis.

Diese Forschungsfrage wurde über eine quantitative Studie mit Hilfe eines zu implementierenden Framework beantwortet. Hierfür werden unterschiedliche Metaheuristiken konzipiert und implementiert. Als prädiktives Verfahren wurde die Robustheitsoptimierung und als reaktives Verfahren die Reparatur eines Zeitplans über eine Kostenfunktion zum Unsicherheitszeitpunkt ausgewählt. Über das Durchlaufen der pro-, prä- und reaktiven Verfahren auf eine gemeinsame Benchmark-Basis und auf eine Vielzahl an zufällig generierten Unsicherheitsszenarien konnten Daten für die quantitative Analyse erhoben werden.

Die Daten für Multi-Mode Resource-Constrained Project Scheduling Problem (MRCPSP) zeigten, dass beide Verfahren Verspätungen durch Aktivitätsstörungen signifikant minimieren, wobei das prädiktive Verfahren insbesondere für kleinere und das reaktive Verfahren für komplexere Unsicherheitsszenarien geeignet ist.

Abstract (english)

The Multi-Mode Resource-Constrained Project Scheduling Problem (MRCPSP) represents an NP-complete optimization problem dealing with the generation of project plans with resource constraints. Uncertainties, such as delays, cause a project plan to fail to complete in planned time.

The goal of the master thesis is first to find baseline schedules for projects using metaheuristics, which should minimize the project duration. Subsequently, predictive and reactive methods have to be selected in order to minimize the associated delays due to uncertainties. „What is the impact of predictive and reactive approaches for MRCPSP based on metaheuristic algorithms in generating schedules in terms of project duration in the presence of delays?“ poses the research question of the thesis.

This research question was answered via a quantitative study using a framework that needed to be implemented. For this purpose, different metaheuristics are designed and implemented. Robustness optimization was selected as the predictive procedure and repair of a schedule via a cost function at the uncertainty time as the reactive procedure. By running the proactive, preactive and reactive procedures on a common benchmark basis and on a variety of randomly generated uncertainty scenarios, data for quantitative analysis could be obtained.

The data of the MRCPSP showed that both methods significantly minimize delays due to activity perturbations, with the predictive method being particularly suitable for smaller uncertainty scenarios and the reactive method for more complex scenarios.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

RCPSP Resource-Constrained Project Scheduling Problem

MRCPSP Multi-Mode Resource-Constrained Project Scheduling Problem

JSP Job-Shop Problem

SGS Schedule Generation Scheme

S-SGS Serial Schedule Generation Scheme

P-SGS Parallel Schedule Generation Scheme

EST Earliest Starting Time

EFT Earliest Finishing Time

LST Latest Starting Time

LFT Latest Finishing Time

RS Random Sampling

BRS Biased Random Sampling

RBRS Regret Based Biased Random Sampling

LS Local Search

TS Tabu Search

SA Simulated Annealing

GA Genetic Algorithm

POJO Plain Old Java Object

Kapitel 1

Einleitung

Dieses Kapitel der Einführung beginnt zunächst mit dem Abschnitt 1.1, welcher das Thema der Masterarbeit motiviert. Im Folgeabschnitt werden die anzustrebenden Ziele der Masterarbeit einschließlich der Forschungsfrage und die dazugehörigen Unterfragen vorgestellt. Der Aufbau der vorliegenden Arbeit wird anschließend im Abschnitt 1.3 erläutert.

1.1. Motivation

„Plans are worthless, but planning is everything!“ [EUPEO58, S. 235] Dwight D. Eisenhower vom 14. November 1957. In dem Zitat vom 34. Präsidenten der USA steckt viel Bedeutung, denn diese Aussage kann dahingehend interpretiert werden, dass in Plänen Ereignisse und Eventualitäten nicht stets berücksichtigt werden. Zu den häufigen Fehlern innerhalb des Projektmanagements gehören Fehlentscheidungen, unklare Abhängigkeiten, Ressourcenknappheiten und vieles weitere [vgl. Nel07, S. 74]. Die Folgen von solchen Fehlern sind nicht zuletzt Verspätungen innerhalb des Zeitplans.

Die ressourcenbeschränkte Projektplanung ist ein Feld aus dem Bereich Operation Management und befasst sich mit der Planung von Projekten mit Ressourcen. Ein Projekt besteht aus einzelnen Aktivitäten, welche wiederum Ressourcen zur Vollendung benötigen. Ein Projekt kann beispielsweise die Entwicklung eines neuen Produktes sein, das Bauen eines Gebäudes, aber auch schon das Backen eines Kuchens. Projekte mit Ressourcenbeschränkungen sind bereits zudem in vielen Unternehmen allgegenwärtig. [vgl. Kel14, S. 1 f.]

Zeitpläne ohne Ressourcenbeschränkungen können über Vor- und Rückwärtskalkulationen erstellt werden [vgl. Kel14, S. 9 f.]. Jedoch stellen die Beschränkungen bei der ressourcenbeschränkten Projektplanung ein Problem dar, da die Ressourcen innerhalb der Aktivitätsausführung durchgehend zur Verfügung stehen müssen,

womit andere Verfahren, wie (Meta-)Heuristiken benötigt werden, um gültige Zeitpläne zu erstellen [vgl. Kel14, S. 11]. Dies stellt das ressourcenbeschränkte Projektplanungsproblem (RCPSP) dar [vgl. Kel14, S. 11], welches als ein NP-hartes Optimierungsproblem angesehen wird [vgl. KH98, S. 2]. Im Mittelpunkt der Masterarbeit steht eine Problemerweiterung des RCPSP, nämlich das Multimodus ressourcenbeschränkte Projektplanungsproblem (MRCPSP) welches Aktivitäten über eine von mehreren zur Verfügung stehenden Alternativen durchlaufen werden muss [vgl. WMY14, S. 596].

Der Umgang mit Unsicherheiten, sei es aufgrund einer Fehlplanung, tatsächlicher Ressourcenknappheiten, Krankheiten innerhalb des Personals oder weiteren Gründen stellt ebenfalls einen wesentlicher Faktor der Masterarbeit dar. Hierfür existieren verschiedene Verfahren, wie die proaktive, prädiktive oder reaktive Planung. Bei der proaktiven Planung werden Unsicherheiten ignoriert. Der prädiktive Ansatz nutzt weitere Kennziffern, wie die Robustheit eines Zeitplans, welche es zu maximieren gilt, um so Verspätungen durch Unsicherheiten zu verringern. Beim reaktiven Ansatz wird zum Zeitpunkt einer Unsicherheit reagiert, um so den Zeitplan zu reparieren oder einen neuen Zeitplan zu erstellen. [vgl. BKF12, S. 404 f.]

Zum Zeitpunkt der Masterarbeit bietet das Multi-Mode Ressource Constrained Project Scheduling Problem (MRCPSP) mit dem Umgang von Unsicherheiten weiterhin ein hohes Forschungspotential. Ein direkter Vergleich von Verfahren für den Umgang mit Unsicherheiten wurde für das MRCPSP noch nicht gezogen. Die Kombination von Metaheuristiken mit den pro-, prä-, reaktiven Methoden gilt es zudem ebenfalls zu evaluieren.

1.2. Problemstellung

Das Problem der Unsicherheiten gilt es für das MRCPSP zu lösen, um so die einhergehenden Verspätungen so gering wie möglich zu halten. Im Rahmen der Masterarbeit wird daher die folgende Forschungsfrage aufgestellt: Welche Auswirkungen haben prädiktive und reaktive Ansätze für das MRCPSP auf Basis von metaheuristischen Algorithmen bei der Erstellung von Zeitplänen in Bezug auf die Projektdauer bei Verspätungen?

Die Forschungsfrage wird mithilfe weiterer Unterfragen gestützt. Welche prädiktiven und reaktiven Ansätze können für das MRCPSP in Kombination mit metaheuristischen Algorithmen angewandt werden? Lässt sich ein Vergleich zwischen prädiktiven und reaktiven Ansätzen auf das Ergebnis in Bezug zur Projektdauer ziehen? Wel-

chen Einfluss haben hierbei Metaheuristiken zur Findung von „guten“ Lösungen?

Um die Unterfragen beantworten zu können müssen die proaktiven, prädiktiven und reaktiven Verfahren auf Benchmarks mit Unsicherheitsszenarien angewandt werden. Diese Benchmarks müssen zunächst für das MRCPSP selektiert werden. Zudem gilt es die Unsicherheitsszenarien innerhalb der Benchmarks zu konzipieren und zu integrieren. Um geeignete Lösungen für das Problem zu finden sollen Metaheuristiken eingesetzt werden. Die Auswahl und die Integrierung, aber auch deren Evaluierung der Metaheuristiken über einen Vergleich der gefundenen Projektdauern sind ebenfalls wesentlicher Bestandteil dieser Arbeit.

1.3. Aufbau der Arbeit

Zu Beginn der Arbeit wurde bereits das Thema der Masterarbeit motiviert und die Ziele innerhalb der Problemstellung definiert. Die folgenden Kapitel der Arbeit befassen sich mit den folgenden Inhalten:

Kapitel 2 In dem Kapitel gilt es die theoretischen Grundlagen des (Multi-Mode) Resource-Constrained Project Scheduling Problem zu erläutern. Sowohl das Grundproblem als auch die Multi-Mode Problemerweiterung werden zunächst formal definiert. Zudem sollen heuristische Lösungsansätze für das MRCPSP vorgestellt werden. Des Weiteren werden in dem Kapitel die theoretischen Grundlagen von einer Auswahl an Metaheuristiken vorgestellt. Zuletzt wird in dem Kapitel der aktuelle Stand der Forschung präsentiert, in welcher das Thema der Masterarbeit eingeordnet wird.

Kapitel 3 Das darauffolgende Kapitel befasst sich mit der Konzeption des MRCPSP-Framework. Hierbei wird im ersten Abschnitt gemäß der Problemstellung basierend auf dem Stand der Forschung ein Konzeptüberblick gegeben. Relevante Teilkomponenten des Konzepts werden im weiteren Verlauf des Kapitels behandelt.

Kapitel 4 Dieses Kapitel basiert auf dem im vorherigen Kapitel vorgestellten Konzept und geht auf die konkrete Umsetzung des MRCPSP-Framework ein. Einleitend wird zunächst die Struktur des Frameworks vorgestellt. Im Anschluss werden einzelne Teilkomponenten erläutert, welche für die Beantwortung der Forschungsfragen relevant sind.

Kapitel 5 Wesentlicher Bestandteil des Kapitels ist der quantitative Vergleich der entwickelten pre-, prä- und reaktiven Verfahren und Metaheuristiken gemäß der Problemstellung.

Kapitel 6 Das abschließende Kapitel 6, bestehend aus dem Fazit und dem Ausblick, beantwortet die Forschungsfrage und dessen Unterfragen anhand der Evaluation und fasst die Ergebnisse der Arbeit zusammen. Themen und Problemstellungen für weiterführende Arbeiten sind dem Ausblick zu entnehmen.

Kapitel 2

Grundlagen

Dieses Kapitel dient zur Einführung der Schlüsselthemen der Masterthesis. Wesentliches Thema der Arbeit stellt das Optimierungsproblem (Multi-Mode) Resource-Constrained Project Scheduling Problem. Dieses (M)RCPSP wird zunächst im Abschnitt 2.1 erläutert. Metaheuristiken stellen eine generische Möglichkeit zur Lösung von Optimierungsproblemen dar. Abschnitt 2.2 zeigt die Funktionsweise von verschiedenen Metaheuristiken auf.

2.1. (Multi-Mode) Resource-Constrained Project Scheduling Problem

Eines der am meisten verbreiteten Optimierungsprobleme stellt das Resource Constrained Project Scheduling Problem dar. Dieses Problem findet insbesondere bei der Projektplanung, aber auch in vielen Bereichen der Gegenwart einen realistischen Bezug. Das Problem bezieht sich auf das Finden von optimalen Schedules (dt. Zeitplänen) von festgelegten Projektablaufplänen, wobei die Aktivitäten erneuerbare Ressourcenanforderungen (z B. Maschinen oder menschliche Arbeitskraft) aufweisen. Optimale Schedules können nicht ohne Weiteres bestimmt werden, folglich müssen für einen gegebenen Projektplan alle Möglichkeiten durchiteriert werden. Dies stellt durch die Komplexität des Lösungsraums ein $\mathcal{NP}$-hartes Problem dar [vgl. KH98, S. 2]. Das RCPSP-Basisproblem wird im Abschnitt 2.1.1 erläutert.

Das RCPSP sieht bei Aktivitäten nur eine Art der Ausführung vor. Zudem berücksichtigt das Grundproblem keine nicht-erneuerbaren Ressourcen, wie z. B. Geldmengen oder Verbrauchsgüter, welche insbesondere in der Praxis und in Projekten wesentliche Bestandteile sind. Diese Aspekte werden in einer Problemerweiterung, nämlich dem Multi-Mode Resource-Constrained Project Scheduling Problem berücksichtigt [vgl. WMY14, S. 596 f.], welche im Abschnitt 2.1.2 beleuchtet werden.

Um zunächst mögliche, wenn auch nicht optimale Lösungen zu erzeugen, wird unter anderem das Verfahren S-SGS eingesetzt. Dieses ermöglicht das Aufbauen von Schedules über das inkrementelle Erweitern von partiellen Schedules [vgl. KH98, S. 3]. S-SGS findet bei Heuristiken, aber auch in Metaheuristiken Verwendung [vgl. KH98, S. 3]. Beleuchtet wird das Konzept einschließlich der gängigen Heuristiken im Abschnitt 2.1.3.

In der Realität sind Unsicherheiten in Zeitplänen nicht abwegig. Diese können beispielsweise in Form von Verspätungen, aber auch über erhöhten Ressourcenverbrauch auftreten. Diese Thematik samt Möglichkeiten zum Umgang des Problems werden im Abschnitt 2.1.4 erläutert.

2.1.1. Resource-Constrained Project Scheduling Problem

Eines der am meisten behandelten Optimierungsprobleme stellt das RCPSP dar. Das grundlegende RCPSP behandelt das Finden von Schedules (dt. Zeitplänen) bei einem gegebenen Projektplan. Ein Projekt besteht aus einer Menge von Aktivitäten $J = \{0, ..., j, j+1\}$, wobei es sich J_0 und J_{j+1} um Dummy-Aktivitäten handeln, welche den Projektstart bzw. das Projektende repräsentieren. Aktivitäten können untereinander in Beziehung stehen, sodass eine Aktivität $j \in J$ erst gestartet werden kann, wenn alle seine Vorgänger $P(j) \in 2^J$ fertiggestellt wurden. Diese besitzen im Grundproblem zudem eine feste Dauer $d_j \in \mathbb{N}_0$ und Ressourcenanforderungen $r_{j,k} \in \mathbb{N}_0$. Bei den Ressourcen wird zwischen erneuerbaren Ressourcenarten $K = \{1, ..., R\}$ unterschieden. Alle Ressourcenarten stehen innerhalb eines Projektes nur in limitierter Anzahl $R_k \in \mathbb{N}$ zur Verfügung, welche während der Projektausführung über die Aktivitäten aufgeteilt werden müssen. Für die Projektstart- und Ende-Aktivitäten gelten zudem $\forall\, k \in K : r_{j,k} = 0$ und $d_j = 0$. Diese Restriktion zeigt auf, dass keine Ressourcen- und Zeitanforderungen für die Aktivitäten gegeben sind. Zudem dürfen Aktivitäten nicht bei der Ausführung unterbrochen werden. [vgl. KH98, S. 1 ff.].

Das Ziel des RCPSP liegt darin, den Zeitplan mit der minimalsten Makespan (zu dt. Projektausführungsdauer) zu bestimmen. Formal wird die Zielfunktion des RCPSP von [KH98, S. 1] vergleichsweise wie folgt beschrieben:

$$\min F_{n+1} = \min C_{max} \qquad \textit{es muss gelten:} \tag{2.1}$$

$$\forall j \in J, h \in P_j : F_h \leq F_j - d_j \tag{2.2}$$

$$\forall k \in K, t \geq 0 : \sum_{j \in A(t)} r_{j,k} \leq R_k \tag{2.3}$$

Die Menge der zum Zeitpunkt t ausgeführten Aktivitäten wird über $A(t)$ angegeben. F_j entspricht der Endzeit einer Aktivität $j \in J$. Eine Aktivität kann erst gestartet werden, wenn alle unmittelbaren Vorgänger abgeschlossen wurden (vgl. Formel 2.2). Formel 2.3 gewährleistet, dass die Kapazitäten von den erneuerbaren Ressourcen zu jedem Zeitpunkt nicht überschritten werden dürfen.

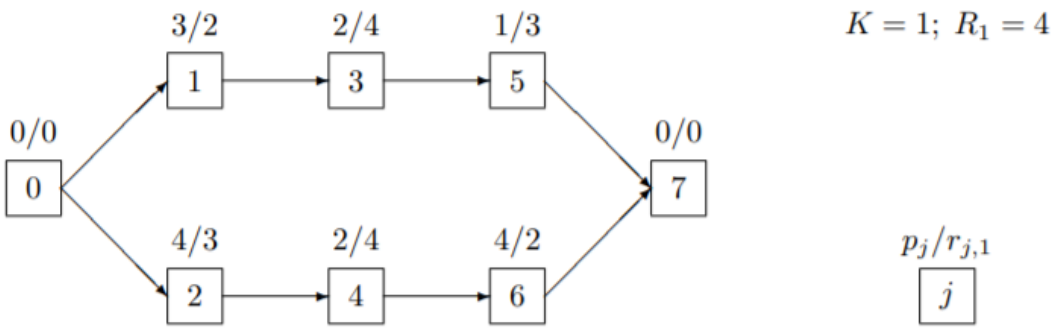


Abbildung 2.1.: Graphen-Darstellung eines RCPSP Beispiel-Projektplans mit $|J| = 8$ Aktivitäten

Quelle: [KH98, S. 2]

Abbildung 2.1 stellt einen beispielhaften Projektplan mit $|J| = 8$ Aktivitäten als Knoten samt deren Ressourcen- und Zeitanforderungen dar. Diese stehen oberhalb der Knoten in der Schreibweise $d_j/r_{r,i}$. J_0 und J_7 entsprechen den Dummyknoten, welche keine Ressourcen- und Zeitanforderungen aufweisen. Die Kanten zeigen Beziehungen zu den Nachfolgeaktivitäten auf. Zudem steht eine erneuerbare Ressourcenart mit einer Kapazität von 4 Mengeneinheiten ($R_1 = 4$) zur Verfügung.

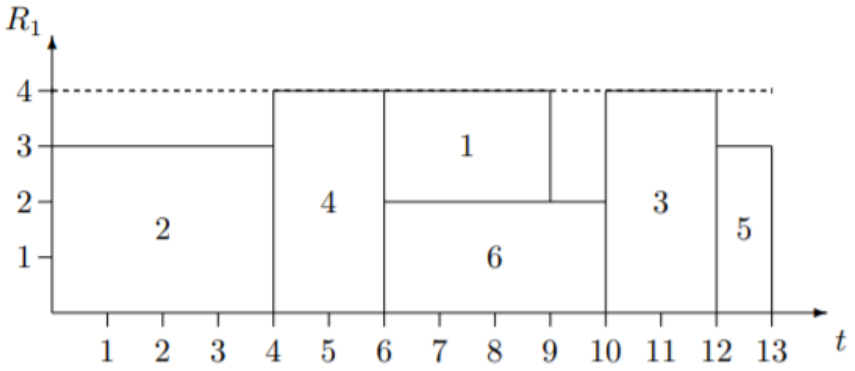


Abbildung 2.2.: Möglicher Zeitplan zum RCPSP Beispiel-Projektplan

Quelle: [KH98, S. 2]

Ein möglicher Zeitplan zu dem Beispiel-Projektplan lässt sich in der Abbildung 2.2 darstellen. Hierbei werden die Ressourcenanforderungen der Aktivitäten mit

Berücksichtigung der Abhängigkeiten auf der Zeitachse gemäß deren Dauer zugeordnet. Die gestrichelte, horizontale Linie entspricht der maximalen Kapazität für die Ressource K_1, welche $R_1 = 4$ gleichkommt. Diese darf bei der Zeitplanung nicht überschritten werden. Der Makespan (dt. Zykluszeit) $C_{max} = 13$ des Zeitplans lässt sich über der Belegung der erneuerbaren Ressourcenarten über den Endzeitpunkt der letzten Aktivität J_{j+1} ablesen.

Das RCPSP wird als eine generalisierte Form des Job-Shop Problem (JSP) angesehen, welches als $\mathcal{NP}$-hartes Problem klassifiziert wurde [vgl. KH98, S. 2]. Konkret bedeutet dies, dass das Finden des besten Zeitplans nur bei einem kleinen Lösungsraum deterministisch über Bruteforcing möglich ist. Bei komplexeren Lösungsräumen sind jedoch zu viele Möglichkeiten vorhanden, als dass in absehbarer Zeit alle Kombinationen betrachtet werden können.

Für das zugrunde liegende Problem wurden (Meta-)Heuristiken und weitere intelligente Verfahren eingeführt, welche zur Findung von adäquaten Zeitplänen beim RCPSP eingesetzt werden können [vgl. KH98, S. 2]. Der Fokus dieser Masterarbeit liegt bei den Metaheuristiken, welche im Abschnitt 2.2 eingeführt werden.

2.1.2. Multi-Mode Problemerweiterung

Das MRCPSP basiert grundlegend auf dem RCPSP und stellt somit eine Erweiterung des Basisproblems dar.

Im MRCPSP steht für jede Aktivität $j \in J$ eine Menge von Modi $m \in M_j = \{1, ..., m_j\}$ zur Verfügung. Hierbei muss eine Aktivität in einem verfügbaren Modus $m \in M_j$ ausgeführt werden. Folglich liegen die Zeit- und Ressourcenanforderungen nicht direkt bei den Aktivitäten, sondern bei den Modi. Die Zeitanforderung einer Aktivität $j \in J$ im Modus $m \in M_j$ wird über $d_{j,m}$ angegeben. Bei den verfügbaren Ressourcenarten kommen neben den erneuerbaren Ressourcenarten $K = \{1, ..., R\}$ auch nicht-erneuerbare Ressourcenarten $L = \{1, ..., C\}$ hinzu, welche über die Modi der Aktivitäten konsumiert werden. Die Menge an nicht-erneuerbaren Ressourcen stehen wie die erneuerbaren Ressourcen nur in limitierter Anzahl $C_l \in \mathbb{N}$ zur Verfügung. Die Ressourcenanforderungen einer Aktivität $j \in J$ im Modus $m \in M_j$ wird bei erneuerbaren Ressourcenarten über $r_{j,m,k}$ und bei nicht-erneuerbaren Ressourcenarten über $c_{j,m,l}$ angegeben. [vgl. WMY14, S. 596 ff.]

Das Ziel des MRCPSP ist identisch mit dem des RCPSP und befasst sich ebenfalls mit dem Finden eines Zeitplans mit der minimalsten Projektausführungsdauer.

Aufgrund der Problemerweiterung gegenüber dem RCPSP sind zudem Anpassungen in der Zielfunktion erforderlich, welche [vgl. WMY14, S. 597] aufführt:

$$\min F_{n+1} = \min C_{max} \qquad \textit{es muss gelten:} \tag{2.4}$$

$$\forall j \in J : \sum_{m \in M_j} x_{j,m} = 1 \tag{2.5}$$

$$\forall j \in J, h \in P_j : F_h \leq F_j - \sum_{m \in M_j} x_{j,m} \cdot d_{j,m} \tag{2.6}$$

$$\forall k \in K, t \geq 0 : \sum_{j \in A(t)} \sum_{m \in M_j} x_{j,m} \cdot r_{j,k} \leq R_k \tag{2.7}$$

$$\forall l \in C, t \geq 0 : \sum_{j \in J} \sum_{m \in M_j} x_{j,m} \cdot c_{j,l} \leq C_l \tag{2.8}$$

$x_{j,m}$ stellt hierbei eine Entscheidungsvariable dar, die angibt, ob ein Modus $m \in M$ für die Aktivität $j \in M$ selektiert wurde. Sofern dies der Fall ist, entspricht $x_{j,m} = 1$, andernfalls $x_{j,m} = 0$. Die Beschränkung in 2.5 nutzt diese Entscheidungsvariable, um sicherzustellen, dass für jede Aktivität immer ein Modus ausgewählt wurde. Die Beschränkung aus 2.6 stellt die Erweiterung aus 2.2 dar und soll weiterhin sicherstellen, dass die Abhängigkeitsbeziehungen zwischen den Aktivitäten für alle ausgewählten Modis eingehalten werden. 2.7 berücksichtigt ebenfalls nun die Modusauswahl und stellt sicher, dass die Kapazität an erneuerbaren Ressourcen zu keinem Zeitpunkt überschritten werden darf. Die Beschränkung aus 2.8 stellt sicher, dass zu keinem Zeitpunkt mehr nicht-erneuerbare Ressourcen konsumiert werden, als zur Verfügung stehen.

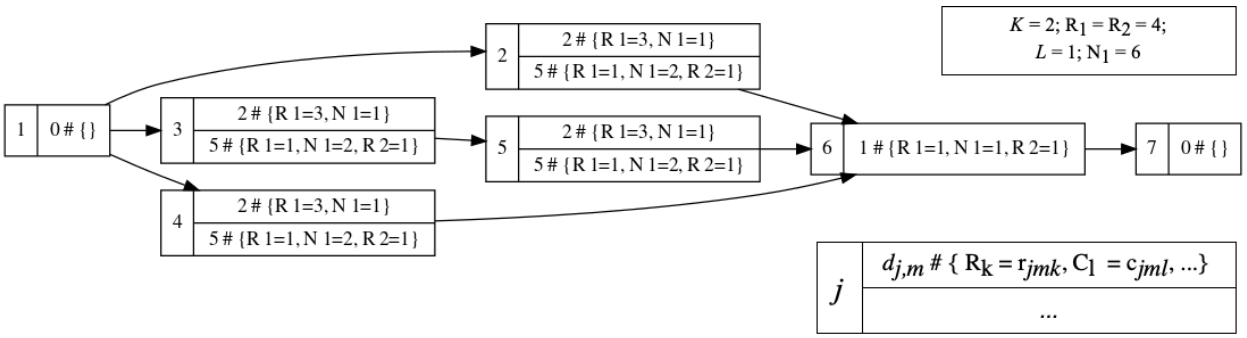


Abbildung 2.3.: Graphen-Darstellung eines MRCPSP Beispiel-Projektplans mit $|J| = 7$ Aktivitäten

Quelle: Eigene Darstellung

Abbildung 2.3 zeigt eine grafische Darstellung eines MRCPSP Beispielprojektes mit 7 Aktivitäten dar, wobei J_0 und J_7 die Dummy-Knoten entsprechen. Innerhalb der Aktivitätsknoten befinden sich zudem die zugehörigen ausführbaren Modi mit

deren Zeit- und Ressourcenanforderungen. Die Aktivitäten $J_2 \dots J_4$ können jeweils in zwei verschiedenen Versionen ausgeführt werden, sofern die Beschränkungen eingehalten werden.

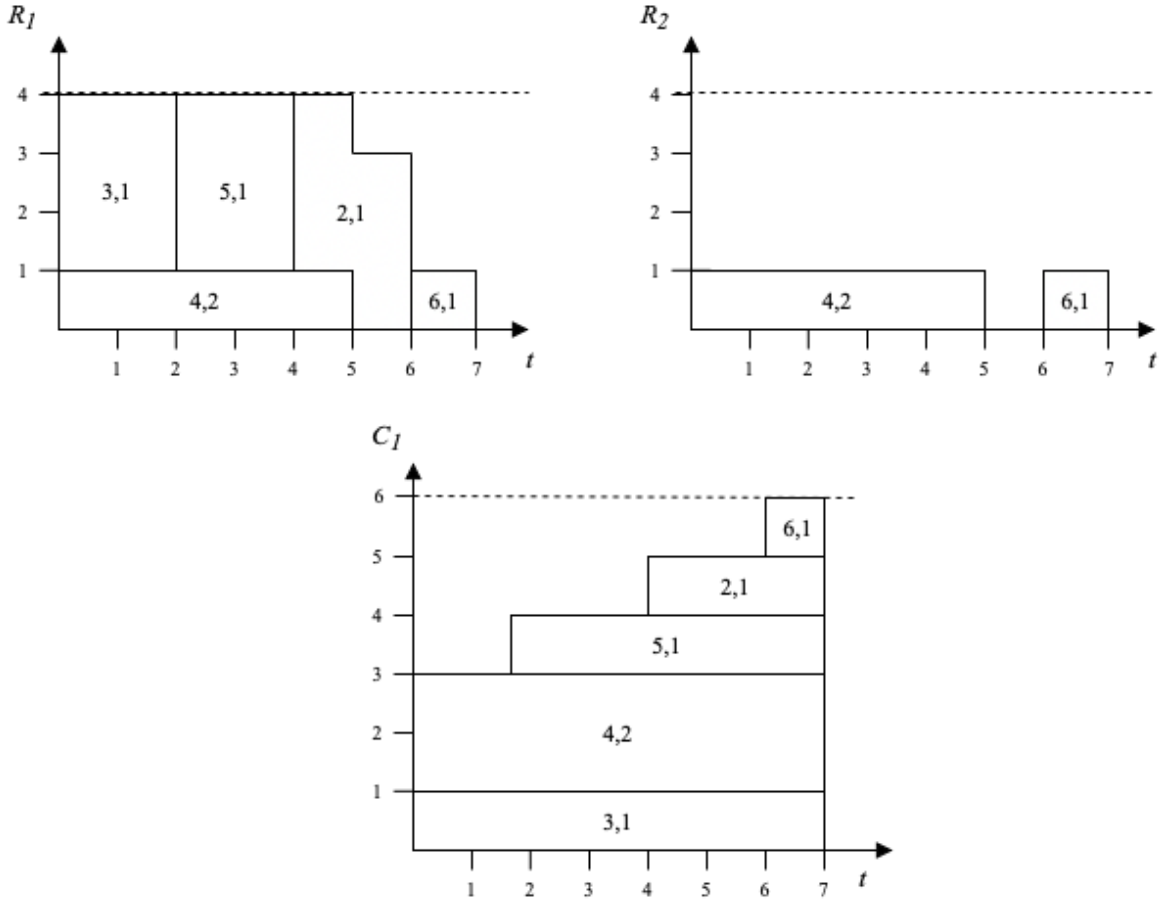


Abbildung 2.4.: Möglicher Zeitplan zum MRCPSP Beispiel-Projektplan

Quelle: Eigene Darstellung

Abbildung 2.4 zeigt für den MRCPSP Beispiel-Projektplan eine optimale Belegung aller Ressourcenarten auf. In der Grafik wurden für alle Aktivitäten auch die zugehörigen Modi gekennzeichnet. Der Makespan für diesen (optimalen) Zeitplan liegt bei $C_{max} = 7$. Dadurch, dass sich nicht-erneuerbare Ressourcen C_l nicht regenerieren, wird dieses Verhalten bei der Ressourcenbelegungsabbildung dahingehend berücksichtigt, dass jede gestartete Aktivität bis zum Projektende andauert. Es lässt sich erkennen, dass für das Beispiel am Projektende keine nicht-erneuerbaren Ressourcen mehr zur Verfügung stehen und folglich das Ausführen von weiteren Aktivitäten über den zweiten Modus zu invaliden Schedules führen würde. Der Verbrauch von nicht-erneuerbaren Ressourcen wäre im zweiten Modus bei allen Aktivitäten im Beispiel-Projektplan höher als bei dem ersten Modus und folglich würde die Grenze C_l überschritten werden.

Aufgrund der nicht-erneuerbaren Ressourcenanforderungen und der Möglichkeit, Aktivitäten über verschiedene Modis auszuführen bewies [KD97, S. 3 f.], dass sich das MRCPSP nicht um ein $\mathcal{NP}$-hartes, sondern sogar um ein $\mathcal{NP}$-vollständiges Problem handelt. Dies ist dann der Fall, wenn mehr als zwei nicht-erneuerbare Ressourcenarten $|L| \geq 2$ und nicht-Dummy Aktivitäten in mehr als zwei Modi $\exists j \in J \,/\, \{J_0, J_{j+1}\} : |M_j| \geq 2$ ausgeführt werden können [vgl. KD97, S. 3 f.]. Durch die Auswahl der Modi wird direkt der Verbrauch an nicht-erneuerbaren Ressourcen beeinflusst. Die nicht-erneuerbaren Ressourcenanforderungen können nach Selektion der Modi die verfügbare Kapazität C_l überschreiten, was ein Schedule gemäß Bedingung 2.8 ungültig macht.

Heuristiken basierend auf prioritätsbasierten Regeln stellen für die Berechnung von Schedules auch beim MRCPSP eine wichtige Technik für das Finden von adäquaten, wenn aber nicht optimalen Lösungen dar. Diese werden im Abschnitt 2.1.3 genauer beleuchtet. Bei komplexen Projekten mit mehr als 20 Aktivitäten kommen Heuristiken an ihre Grenzen. Folglich werden Metaheuristiken eingesetzt, welche mittels lokaler Suche versuchen die optimale Lösung zu finden. Viele Metaheuristiken benötigen initiale Lösungen, wofür Heuristiken jedoch eine gute Basis darstellen. [vgl. LTB06, S. 69 f.]

2.1.3. Serial Schedule Generation Scheme

Schedule Generation Schemes (SGSs) stellen für das (M)RCPSP wesentliche Techniken zur Erzeugung von Zeitplänen dar. Die am häufigsten in der Literatur verwendeten SGSs sind Serial Schedule Generation Schemes (S-SGSs) und Parallel Schedule Generation Schemes (P-SGSs). Das Prinzip eines SGS liegt darin, vollständige Zeitpläne inkrementell über partielle Zeitpläne aufzubauen. Ein partieller Schedule entspricht einem Zeitplan, welcher unvollständig ist und nicht alle Aktivitäten berücksichtigt, sondern nur eine Teilmenge von Aktivitäten. [vgl. KH98, S. 3 f.]

Die beiden SGS Varianten unterscheiden sich bei der Inkrementierungsweise voneinander. Beim S-SGS werden n Phasen durchlaufen, wobei in jeder Phase eine Aktivität selektiert wird. Bei dem P-SGS werden bis zu n Phasen durchlaufen. Zu jedem Aktivitätsendzeitpunkt wird überprüft, welche Aktivitäten ausgewählt werden können. Während das S-SGS aktivitätsinkremental funktioniert, agiert das P-SGS zeitinkremental. [vgl. KH98, S. 3 f.]

Im Rahmen der Masterthesis wird vermehrt das MRCPSP behandelt, welches in Kombination mit dem Parallel Schedule Generation Scheme (P-SGS) nicht immer

eine optimale Lösung darstellt [vgl. AMR03, S. 911 f.] [vgl. KH98, S. 5 f.] . Folglich beschränkt sich diese Arbeit auf das S-SGS.

Listing 2.1: S-SGS-Algorithmus (Quelle: [KH98, S. 3])

1 Init: F_0, $S_0 = \{0\}$,
2 **for** g = 1 to n **do**
3 Berechne $\mathcal{D}_g, \mathcal{F}_g, \tilde{\mathcal{R}}_k(t)\,(k \in \mathcal{K}; t \in \mathcal{F}_g)$
4 Wähle ein $j \in \mathcal{D}$ aus
5 $EFT_j = \max_{h \in \mathcal{P}_j}\{F_h\} + p_j$
6 $F_j = \min\{t \in [EFT_j - p_j, LFT_j - p_j] \cap \mathcal{F}_g | r_{j,k} \leq R_k(\tau), k \in \mathcal{K}, \tau \in [t, t + p_j[\cap \mathcal{F}_g\} + p_j$
7 $\mathcal{S}_g = \mathcal{S}_{g-1} \cup \{j\}$
8 $F_{n+1} = \max_{h \in \mathcal{P}_{n+1}}\{F_h\}$

Listing 2.1 stellt den S-SGS-Algorithmus aus dem Manuskript von [KH98] dar. Hierbei wird in jeder Phase g die Mengen $\mathcal{S}_g$ und $\mathcal{D}_g$ berechnet. Die Menge $\mathcal{S}_g$ beinhaltet bereits berücksichtigte Aktivitäten, während $\mathcal{D}_g$ die selektierbaren Aktivitäten für die Phase g beinhaltet. Die Menge $\mathcal{F}_g$ beinhaltet zu $\mathcal{S}_g$ die Endzeiten der einzelnen Aktivitäten. $\tilde{\mathcal{R}}_k(t)$ gibt die Anzahl noch verfügbarer erneuerbarer Ressourcen für die Ressourcenart k zum Zeitpunkt t an. [vgl. KH98, S. 3 f.]

Abbildung 2.5 zeigt auf, wie S-SGS auf einen Projektplan angewendet werden kann, um eine mögliche Lösung für das RCPSP zu erhalten. Die Auswahl der Aktivität j aus $\mathcal{D}_g$ erfolgte in dem Beispiel willkürlich.

g	1	2	3	4	5	6
$\mathcal{D}_g$	{1,2}	{1,4}	{1,6}	{3,6}	{3}	{5}
j	2	4	1	6	3	5

Abbildung 2.5.: S-SGS-Anwendung auf den RCPSP Beispiel-Projektplan von Abbildung 2.1 zur Erstellung des Zeitplans aus Abbildung 2.2

Quelle: [vgl. KH98, S. 3]

Aus dem Listing 2.1 heraus lassen sich weitere Kennziffern von Aktivitäten innerhalb eines Projektplans erkennen. Im Algorithmus wurden Earliest Finishing Time (EFT) und Latest Finishing Time (LFT) verwendet. Diese und Earliest Starting Time (EST) und Latest Starting Time (LST) sind wesentlich für viele prioritätsbasierte Aktivitätsregeln (vgl. Abschnitt 2.1.3.2) und für die Robustheitsberechnungen (vgl. Abschnitt 2.1.4.1) und werden in der Tabelle 2.1 eingeführt.

Abk.	Beschreibung [vgl. Bur99, S. 126]	Formel [vgl. Bur99, S. 127 ff.]
EST	Earliest Starting Time (dt. frühster Startzeitpunkt) stellt den frühstmöglichen Startzeitpunkt einer Aktivität j dar.	$EST_j = \max_{h \in P(j)}\{EFT(h)\}$
EFT	Earliest Finishing Time (dt. frühster Endzeitpunkt) stellt den frühstmöglichen Endzeitpunkt einer Aktivität j dar.	$EFT_j = EST_j + d_j$
LST	Latest Starting Time (dt. spätester Startzeitpunkt) stellt den spätesten Startzeitpunkt einer Aktivität j dar, ohne dass das Projekt in Verzug kommt.	$LST_j = LFT_j - d_j$
LFT	Latest Finishing Time (dt. spätester Endzeitpunkt) stellt den spätesten Endzeitpunkt einer Aktivität j dar, ohne dass das Projekt in Verzug kommt.	$LFT_j = \min_{h \in P(j)}\{LST(h)\}$

Tabelle 2.1.: Kennziffern der kritischen Pfadmethode

Quelle: In Anlehnung an [Bur99]

2.1.3.1. Aktivitäts- und Moduslisten

Aktivitäts- und Moduslisten sind wesentliche Techniken zur Repräsentation von Lösungen für das (M)RCPSP. Insbesondere Metaheuristiken nutzen diese Art der Repräsentation, um aus den Listen Zeitpläne zu transferieren [vgl. RSMA15, S. 10], [vgl. WMY14, S. 602], [vgl. LG13, S. 2370].

Zeitpläne im RCPSP lassen sich über Aktivitätslisten $\lambda = \langle j_1, j_2, ..., j_n \rangle$ darstellen. Die Reihenfolge der Aktivitäten entspricht die der Planungsreihenfolge. Folglich müssen die Abhängigkeiten der Aktivitäten zueinander innerhalb der Aktivitätsliste λ eingehalten werden. Im Beispiel der Abbildung 2.2 entspricht die Aktivitätsliste $\lambda = \langle 2, 4, 1, 6, 3, 5 \rangle$. [vgl. KH98, S. 3 f.]

Für das MRCPSP müssen zudem Modi der zugehörigen Aktivitäten berücksichtigt werden. Hierfür werden zusätzlich zu den Aktivitätslisten auch Moduslisten eingesetzt. Eine Modusliste $\mu = \langle m_1, m_2, ..., m_n \rangle$ berücksichtigt ebenfalls die Planungs-

reihenfolge. Die Länge der Modusliste entspricht der Länge der Aktivitätsliste $|\mu| \equiv |\lambda|$. Folglich können über Aktivitäts- und Moduslisten die Aktivitäten und Modi über die Positionierung zugeordnet werden [vgl. SAA15, S. 908]. Für das Beispiel aus Abbildung 2.4 entspricht die Aktivitäts- und Modusliste das Tupel $(\lambda, \mu) = (\langle 1, 3, 4, 5, 2, 6, 7\rangle, \langle 1, 1, 2, 1, 1, 1, 1\rangle)$. Abbildung 2.6 illustriert dieses Beispiel.

λ:	1	3	4	5	2	6	7
μ:	1	1	2	1	1	1	1

Abbildung 2.6.: Darstellung der Aktivitäts- und Modusliste (λ, μ) für das MRCPSP Beispiel-Projektplan

Quelle: Eigene Darstellung

Eine Lösung für das MRCPSP entspricht dem Tupel $I = (\lambda, \mu)$. Für jeden Projektplan existiert mindestens eine optimale Lösung mit der kürzesten Projektdauer $I^* = (\lambda^*, \mu^*)$ [vgl. KH98, S. 4]. Das Ziel der (Meta-)Heuristiken liegt darin, die optimale Lösung I^* zu finden oder dieser zumindest sehr nahezukommen. Hierfür müssen Aktivitäts- und Moduslisten ausgewählt, evaluiert und anschließend weitergesucht werden. Abschnitt 2.1.3.2 befasst sich mit Heuristiken zum Finden von Aktivitätslisten, Abschnitt 2.1.3.3 mit dem Finden von Moduslisten.

2.1.3.2. Prioritätsbasierte Regeln für Aktivitäten

Das Serial Schedule Generation Scheme stellt eine bedeutende Technik zur Erzeugung von Aktivitätslisten dar. Hierbei wird in jeder Phase g eine Aktivität j aus den möglichen Aktivitäten $\mathcal{D}_g$ ausgewählt. Die Auswahl der Aktivität $j \in \mathcal{D}_g$ gilt es mittels prioritätsbasierten Regeln zu bestimmen. [vgl. KH98, S. 5]

Prioritätsbasierte Regeln bestehen aus zwei Komponenten, nämlich einer Prioritätswertfunktion $v(j)$ und einem Auswahlparameter $extr$. Die Prioritätswertfunktion für Aktivitäten ist über $v : \mathcal{D}_g \to \mathbb{R}_0$ gemappt und wird über die Prioritätsregeln definiert. Die Funktion gibt für eine mögliche Aktivität $j \in \mathcal{D}_g$ den entsprechenden Prioritätswert gemäß der Regel an. Der Auswahlparameter $extr \in \{\text{MIN}, \text{MAX}\}$ bestimmt das Extremum. In jeder Phase wird eine Aktivität ausgewählt, die mit dem zugehörigen Prioritätswert dem Extremum aller möglichen Aktivitäten entspricht. Sofern mehrere Aktivitäten dem Extremwert entsprechen, muss eine weitere Heuristik eingeführt werden, wie die Auswahl über die geringste Aktivitätsnummer. [vgl. SR97, S. 6 f.]

Abkürzung	Name	Prioritätsregel-funktion $v(j)$	Auswahl-parameter $extr$
GRPW	Greatest Rank Positional Weight	$d_j + \sum_{i \in S(j)} d_i$	MAX
LFT	Latest Finish Time	LFT_j	MIN
LST	Latest Start Time	LST_j	MIN
MSLK	Minimum Slack	$LFT_j - EFT_j$	MIN
MTS	Most Total Successors	$\|S(j)\|$	MAX

Tabelle 2.2.: Prioritätsregeln für Aktivitäten

Quelle: [vgl. KH98, S. 6]

Die in Tabelle 2.2 aufgeführten etabilierten Regeln werden genutzt, um anhand der Prioritätsregelfunktion $v(j)$ gemäß Auswahlparameter $extr$ die zugehörige Aktivität j auszuwählen. Tabelle 2.3 zeigt ein Beispiel bei Anwendung einer beliebigen Prioritätsregelfunktion $v(j)$ auf. Sofern der Auswahlparameter $extr = \text{MAX}$ entspricht, wird in der Phase g die Aktivität 3 ausgewählt, bei $extr = \text{MIN}$ die Aktivität 2.

$j \in \mathcal{D}_g$	1	2	3	4
$v(j)$	5	3	12	4

Tabelle 2.3.: Beispiele von Prioritätswerten von den möglichen Aktivitäten $j \in \mathcal{D}_g$ einer Phase g

Quelle: Eigene Darstellung

2.1.3.3. Selektionsregeln für Modi

Um beim MRCPSP Zeitpläne zu erzeugen, müssen neben den Aktivitätslisten auch Moduslisten generiert werden. Des Weiteren besteht beim MRCPSP die Problematik, dass bei Anwendung der Prioritätsregeln für eine Aktivität mehrere Modi mit unterschiedlichen Ressourcen- und Zeitanforderungen zur Verfügung stehen. Folglich werden Selektionsregeln für Modi benötigt, welche analog zu den Prioritätsregeln für Aktivitäten funktionieren.

Abkürzung	Name	Selektionsregel-funktion $s(j,m)$	Auswahl-parameter *extr*
LPSRD	Least Product Sum of Res. and Dur.	$\sum_{l=1}^{\|L\|}(c_{j,m,l} \cdot d_{j,m})$	MIN
LRP	Least Resource Proportion	$\max(\frac{r_{j,m,k}}{\|K\|})$	MIN
LRS	Least Sum of Non-renewable Resource	$\sum_{l=1}^{\|L\|} \frac{c_{j,m,l}}{R_k^v}$	MIN
LTRU	Least Total Resource Usage	$\sum_{l=1}^{\|L\|} c_{j,m,l}$	MIN
SFM	Shortest Feasible Mode	$d_{j,m}$	MIN

Tabelle 2.4.: Selektionsregeln für Modi

Quelle: [vgl. CLP14, S. 5046]

Die in Tabelle 2.4 aufgeführten etabilierten Regeln werden genutzt, um anhand der Selektionsregelfunktion $s(j,m)$ gemäß Auswahlparameter *extr* den zugehörigen Modus $m \in M_j$ einer Aktivität j auszuwählen. Tabelle 2.5 zeigt ein Beispiel bei Anwendung einer beliebigen Prioritätsregelfunktion $v(j)$ und Selektionsregelfunktion $s(j,m)$ auf. Um die Aktivitätsregelfunktion anwenden zu können, müssen vorher die Modi selektiert werden [vgl. CLP14, S. 5046]. Die Selektion der Modi und Aktivitäten geschieht im Beispiel über $extr = \text{MAX}$. Für alle möglichen Aktivitäten der Phase $j \in \mathcal{D}_g$ wird somit der erste Modus ausgewählt. Im Anschluss können die Aktivitäten über die Prioritätsregeln ausgewählt werden (vgl. Abschnitt 2.1.3.2).

$j \in \mathcal{D}_g$	1		2		3		4	
$m \in M_j$	1	2	1	2	1	2	1	2
$s(j,m)$	5	3	2	1	5	4	7	7
$v(j)$	5		3		12		2	

Tabelle 2.5.: Beispiele von Prioritäts- und Selektionswerten von den möglichen Aktivitäten $j \in \mathcal{D}_g$ und Modis der Aktivitäten einer Phase g

Quelle: Eigene Darstellung

2.1.3.4. Sampling als Multi Pass Methode

Das Anwenden der Aktivitäts- und Modusregeln führt innerhalb eines S-SGS zu einem einzigen Zeitplan und wird somit als eine Single Pass Methode klassifiziert [vgl. KH98, S. 6]. Das Anwenden dieser Heuristiken führt bei gleichbleibendem Input zudem immer zu identischen Ergebnissen und sind folglich deterministisch [vgl. SR97, S. 4]. Dennoch besteht das Interesse eine breitere Menge an möglichen Zeitplänen zu untersuchen, um durchaus bessere Ergebnisse als das direkte Anwenden der Regeln zu erhalten.

Unter Multi Pass Methoden werden wiederum Verfahren bezeichnet, die in der

Lage sind, mehrere Zeitpläne zu generieren. Hierunter fallen Verfahren, wie das Anwenden von mehreren Regeln, aber auch das Sampling. Für jede mögliche Aktivität einer Phase $j \in \mathcal{D}_g$ und deren jeweiligen Modi $m \in M_j$ ist hierbei ein Wahrscheinlichkeitswert gemäß Funktion $p(x) \in [0, 1]$ vorgesehen, welche für die Entscheidungsmenge aufkommuliert stets 1 ergeben muss. Aus der Entscheidungsmenge wird im Anschluss gemäß der Wahrscheinlichkeit die Aktivität bzw. der Modus ausgewählt. Die Wahrscheinlichkeitsfunktion $p(x)$ wird anhand von Sampling Methoden definiert. [vgl. KH98, S. 7 f.] Aufgrund der Einfachheit beziehen sich die Formeln der folgenden vorgestellten Sampling Methoden nur auf die Aktivitätsauswahl. Diese lassen sich jedoch ohne weiteres auch auf die Modusauswahl ableiten.

Die einfachste Sampling Methode ist Random Sampling (RS). Hierbei ist die Wahrscheinlichkeit $p(x)$ innerhalb der Entscheidungsmenge gleichverteilt $p(j) = \frac{1}{|\mathcal{D}_g|}$. [vgl. KH98, S. 7] Hierbei werden Prioritäts- und Selektionregeln ignoriert und gänzlich zufällige Zeitpläne erzielt.

Eine weitere Sampling Methode ist Biased Random Sampling (BRS). Die Wahrscheinlichkeit $p(x)$ hängt hierbei direkt von der Prioritäts- und Selektionsregel ab $p(j) = \frac{v(j)}{\sum_{i \in \mathcal{D}_g} v(i)}$. [vgl. KH98, S. 7]

Über Regret Based Biased Random Sampling (RBRS) ist es zudem möglich, die Wahrscheinlichkeit $p(x)$ über einen Regretwert (dt. Reuewert) zu beeinflussen [vgl. SR97, S. 12]. Dies führt dazu, dass die Aktivitäts- und Moduslisten gemäß der Prioritäts- und Selektionsregeln mehr oder weniger variieren können. Die Wahrscheinlichkeit $p(x)$ lässt sich über die folgenden Funktionen gemäß [vgl. SR97, S. 12] definieren:

$$v'(j) = \begin{cases} \max_{i \in \mathcal{D}_n}(v(i)) - v(j) & \text{wenn extr} = \min \\ v(j) - \min_{i \in \mathcal{D}_n}(v(i)) & \text{wenn extr} = \max \end{cases} \tag{2.9}$$

$$v''(j) = (v'(j) + \epsilon)^\alpha \tag{2.10}$$

$$p(j) = \frac{v''(j)}{\sum_{i \in D_n} v''(i)} \tag{2.11}$$

$v'(j)$ und $v''(j)$ stellen die Regrets dar. Beim RBRS werden die Regrets zudem mit $\epsilon = \mathbb{R}_{>0}$ und $\alpha = \mathbb{R}_{\geq 0}$ parametrisiert. ϵ garantiert, dass $v''(j) \neq 0$. Dies ist relevant, um innerhalb der Wahrscheinlichkeitsfunktion $p(x)$ Divisionen durch 0 zu vermeiden. α regelt den Einfluss der Prioritäts- und Selektionsregelfunktion auf den Wahrscheinlichkeitswert. Bei einem hohen Wert verstärkt sich der Einfluss, bei einem niedrigen Wert verringert sich dieser. Bei $\alpha = 0$ ist die Prioritäts- und Selekti-

onsregelfunktion irrelevant und folglich sind die Wahrscheinlichkeiten innerhalb der Entscheidungsmenge gleichverteilt, identisch zum RS. Bei $\alpha = \infty$ sind die Wahrscheinlichkeiten so angeordnet, dass die Aktivität oder der Modus deterministisch ausgewählt werden kann, identisch zur Single Pass Methode. [vgl. SR97, S. 12 f.]

Die Funktionsweise der vorgestellten Multi Pass Samplingverfahren lassen sich in Tabelle 2.6 aufbauend auf dem Beispiel aus Abschnitt 2.1.3.2 verdeutlichen. Für dieses Beispiel liegt der Extremwert für die zu betrachtende Prioritätsregel bei $extr$ = MAX. Sofern RBRS als Sampling-Methode ausgewählt wurde, so wird z. B. die Aktivität 3 mit einer 80 prozentigen Wahrscheinlichkeit selektiert. Wenn $\alpha = \infty$, dann wird das Ergebnis gemäß der regelbasierten Single Pass Methode deterministisch ausgewählt.

$j \in \mathcal{D}_g$	1	2	3	4
$v(j)$	5	3	12	4
Random Sampling (RS)	0.25	0.25	0.25	0.25
Biased Random Sampling (BRS)	0,21	0,13	0,50	0,16
Regret Based Biased Random Sampling (RBRS)	0,10	0,03	0,80	0,07
RBRS mit $\alpha = 0$	0.25	0.25	0.25	0.25
RBRS mit $\alpha = \infty$	0.00	0.00	1.00	0.00

Tabelle 2.6.: Selektionswahrscheinlichkeiten $p(j)$ für die vorgestellten Sampling-Verfahren mit $|\mathcal{D}_g| = 4$ und $extr$ = MAX

Quelle: In Anlehnung an [vgl. KH98, S. 8]

2.1.4. Unsicherheiten

Insbesondere in der Realität ist es möglich, dass Unsicherheiten auftreten können. Dies kann der Fall sein, wenn Events, wie der Ausfall von Maschinen, die Erkrankung von Mitarbeitern, die Fehleinschätzung bei der Planung usw. auftreten [vgl. DDH11, S. 64 ff.]. Durch Verspätungen können Projekte in Verzug kommen und somit die geplante Makespan C_{max} überschreiten. Dies ist suboptimal und stellt eine Herausforderung bei der Erzeugung von Zeitplänen dar. Mit solchen Unsicherheiten gilt es auf abstrakter Ebene innerhalb des (M)RCPSP umzugehen.

Es können verschiedene Arten von Unsicherheiten betrachtet werden. Die in der Literatur am häufigsten aufzufindenden Unsicherheitsarten sind Verspätungen von Aktivitäten und erneuerbaren Ressourcen, die nicht zu jedem Zeitpunkt konstant vorliegen. [vgl. DDH11, S. 64 ff.]

Störungen der Aktivitätsdauer treten auf, wenn die tatsächliche Dauer einer Aktivität größer ist, als die geplante Dauer d_j. Die Differenz zwischen geplanter und tatsächlicher Dauer wird über $\Delta_j \in \mathbb{N}$ ausgedrückt. [vgl. DDH11, S. 65] Dies wird in Abbildung 2.7 anhand des MRCPSP Beispiel-Zeitplans aus Abschnitt 2.1.2 bei der Aktivität 4 mit $\Delta_4 = 1$ deutlich. Zudem lässt sich erkennen, dass der geplante Makespan gegenüber dem initialen Zeitplan (Abbildung 2.4) trotz der Störung nicht in Vorzug kommt, da die Verspätung über den Puffer von einer Zeiteinheit und einer verfügbaren Ressource R_1 und mehreren Ressourcen R_2 vor Durchlauf der Folgeaktivität bedient werden konnte.

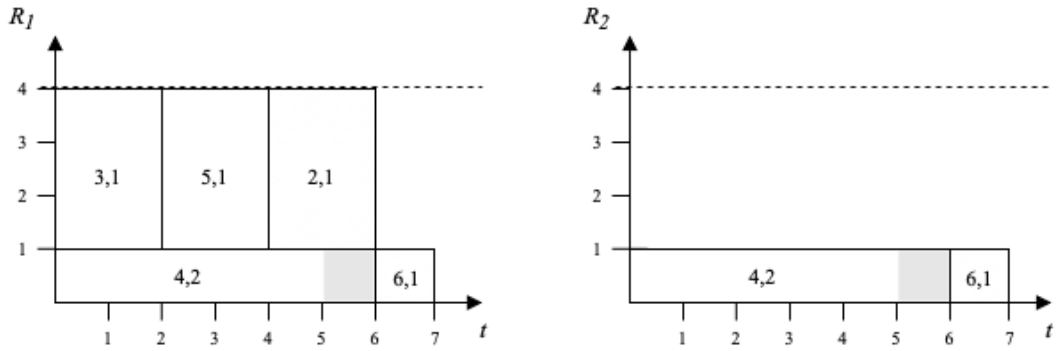


Abbildung 2.7.: MRCPSP Beispiel-Zeitplan mit einer Aktivitätsstörung von $\Delta_4 = 1$

Quelle: Eigene Darstellung

Eine weitere Art von Unsicherheiten stellen Störungen bei den erneuerbaren Ressourcen dar. Bei dieser Störungsart besteht die Annahme, dass die Anzahl der verfügbaren Ressourcen einer Ressourcenart R_k zu einem Zeitpunkt t über die Differenz $\Delta_k^\rho \in \mathbb{N}$ reduziert wird. Über $t + \delta_t$ wird angegeben, ab welchem Zeitpunkt die erneuerbaren Ressourcenarten wieder komplett zur Verfügung stehen [vgl. DDH11, S. 65 f.]. Abbildung 2.8 verdeutlicht die erneuerbaren Ressourcenstörungen. Im Zeitintervall $[4, 7]$ befindet sich eine erneuerbare Ressourcenstörung $\Delta_1^\rho = 1$. Aktivität 2 kann somit nicht planmäßig ab Zeiteinheit 4 gestartet werden, da durch die Störung nicht genügend erneuerbare Ressourcen für den ausgewählten Modus vorhanden sind. Somit muss entweder ein anderer Modus ausgewählt werden oder solange mit dem Start von Aktivität 2 gewartet werden, bis die erneuerbaren Ressourcen für die komplette Laufzeit der Aktivität wieder zur Verfügung stehen. Der Start der Ausführung wäre bei dem selektierten Modus ab $t = 5$ der Fall, da durch das Fertigstellen von Aktivität 4 eine erneuerbare Ressource R_1 freigegeben wird und somit die erforderliche Anzahl von Ressourcen ($r_{2,1,1} = 3$) zur Verfügung stehen. Dadurch kommt jedoch bei Aktivität 6 ein Startverzug von einer Zeiteinheit. Durch diese Störung erhöht sich der Makespan des Zeitplans aus Abbildung 2.8 ($C_{max} = 8$)

gegenüber dem Zeitplan aus Abbildung 2.4 ($C_{max} = 7$) um eine Zeiteinheit.

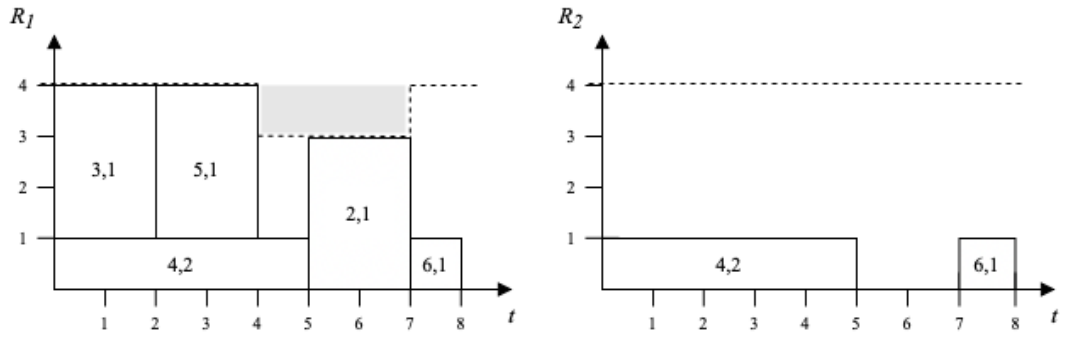


Abbildung 2.8.: MRCPSP Beispiel-Zeitplan mit einer (erneuerbaren) Ressourcenstörung ab $t = 4$ von $\Delta_1^\rho = 1$, welche ab $t + \delta_t = 7$ normalisiert wird

Quelle: Eigene Darstellung

Das Ziel des (M)RCPSP besteht darin, den Zeitplan mit der geringsten Projektausführungsdauer zu finden. Der Umgang mit Unsicherheiten stellt zudem eine Herausforderung dar. [BKF12] beleuchtete in einer Erhebung einige Möglichkeiten zum Umgang des Unsicherheitsproblems für die stochastische Version des RCPSP.

Die naivste Herangehensweise mit Unsicherheiten umzugehen ist es diese bei der Zeitplanerstellung zu ignorieren. Diese Methode wird von [BKF12] als prädiktiv bezeichnet. Das Ignorieren von Unsicherheiten kann jedoch dazu führen, dass der Zeitplan durch Verspätungen stärker in Verzug kommt als bei anderen Herangehensweisen. [vgl. BKF12, S. 404]

Die proaktive Herangehensweise behandelt das Problem der Unsicherheiten, indem eine weitere Zielfunktion, nämlich die Robustheit eingeführt wird [vgl. BKF12, S. 404]. Die Robustheit gibt an, dass der Zeitplan mit geringfügigen Veränderungen (z. B. die der Unsicherheitszenarien) von Aktivitäten umgehen kann [vgl. KC13, S. 246].

Die dritte Herangehensweise stellt die reaktiven Methoden dar. Diese reagieren direkt zum Zeitpunkt der Unsicherheit. Hierbei besteht die Möglichkeit, den Zeitplan ab der Unsicherheitsstelle zu reparieren oder gänzlich ab der Stelle der Unsicherheit einen neuen Zeitplan zu erstellen [vgl. BKF12, S. 404 f.].

In der Literatur werden prädiktive Verfahren auch als proaktive Verfahren bezeichnet und umgekehrt [vgl. KC13, S. 246]. Im Rahmen der Masterarbeit werden folglich die eingangs vorgestellten proaktiven Verfahren nun als prädiktive Verfah-

ren behandelt, welche z. B. die Robustheit optimieren. Die eingangs vorgestellten prädiktiven Verfahren werden somit als proaktive Verfahren angesehen, welche z. B. Unsicherheiten ignorieren. Im Abschnitt 2.1.4.1 werden die prädiktiven Methoden, repräsentierend durch die Robustheit,s konkreter vorgestellt. Die reaktiven Methoden, repräsentierend durch das Erzeugen von neuen Zeitplänen zu den Unsicherheitszeitpunkten, werden im Abschnitt 2.1.4.2 vorgestellt.

2.1.4.1. Prädiktive Methoden

Eine Möglichkeit mit Unsicherheiten umzugehen stellt das vorausschauende Erstellen von Zeitplänen dar. Hierbei wird neben der Minimierung der Projektdauer C_{max} eine weitere Zielfunktion eingeführt, nämlich die Maximierung der Robustheit Ω. Die Robustheit gilt als die Maßeinheit, wie gut mit kleinen Änderungen innerhalb eines Zeitplans umgegangen werden kann, ohne dass die Projektdauer durch die Verspätung ansteigt [vgl. KC13, S. 246] [vgl. AFH05, S. 177].

Eine mögliche Messung von Robustheit kann über die Summe aller freien Puffer der Aktivitäten definiert werden. Unter einem freien Puffer s_j werden die Zeiteinheiten verstanden, die eingesetzt werden können, damit die unmittelbare Folgeaktivität nicht verzögert wird. Die Einhaltung der Ressourcenbeschränkung muss im Puffer dennoch berücksichtigt werden. Folglich ist die Robustheit über $\Omega = \sum_{j=1}^{n} s_j$ definiert. [vgl. AFH05, S. 177]

Der freie Puffer lässt sich über die frühsten und spätesten Start- bzw. Endzeitpunkte berechnen $s_j = LST_j - EST_j \Leftrightarrow s_j = LFT_j - EFT_j$. Während EST_j und EFT_j sich über den erstellten Zeitplan ablesen lassen, müssen LST_j und LFT_j über die Backward Recursive Prodecure berechnet werden [vgl. AFH05, S. 181].

Bei dem RCPSP Beispiel-Zeitplan S^1 aus Abbildung 2.2 liegt die Robustheit bei $\Omega^1 = \sum_{j=1}^{n} s_j = 1$, da der einzige freie Puffer bei Aktivität j_1 liegt. Bei einer Verspätung von $\Delta_1 = 1$ würde es zu keiner Verspätung der Projektdauer C_{max} kommen, da der Puffer genutzt werden kann. Sofern es bei einer anderen Aktivität zu einer Verspätung kommt, wird sich die Projektdauer C_{max} verzögern. Ähnlich sieht es bei dem MRCPSP Beispiel-Zeitplan S^2 aus Abbildung 2.4 aus, welcher ebenfalls einen Wert von $\Omega^2 = \sum_{j=1}^{n} s_j = s_4 = 1$ aufweist. Bei Nutzung des Puffers liegt nach dem Verspätungsszenario $\Delta_4 = 1$ die Robustheit bei $\Omega^{2'} = 0$ (vgl. Abbildung 2.7).

Ein Zeitplan mit der gleichen Projektdauer C_{max} kann unterschiedliche Robustheitswerte aufweisen und ist folglich für Unsicherheitsszenarien mehr oder weniger geeignet [vgl. AFH05, S. 178]. Beim prädiktiven Ansatz gilt es bei Zeitplänen mit

identischer Projektdauer den mit dem höchsten Robustheitswert auszuwählen.

Die Robustheit lässt sich nicht nur über die Summe aller freien Puffer eines Zeitplans definieren. [KC13] vergleicht in seiner Publikation eine Menge von Robustmessungsfunktionen für das RCPSP und stellt das Prinzip der weighted slack functions (dt. gewichtete Pufferfunktionen) vor. Die Slack Function (dt. Pufferfunktion) stellt zum Beispiel die Summe der freien Puffer oder auch einen binärer Puffer dar [vgl. KC13, S. 253 f.]. Der Puffer kann anschließend über ein Weight (dt. Gewicht) multipliziert wird. Folglich wirkt sich ein Puffer für dominierende Aktivitäten stärker gegenüber einfachen Aktivitäten aus. Ein Weight kann die Anzahl von notwendigen Ressourcen, direkten oder totalen Nachfolgern einer Aktivität uvw. darstellen [vgl. KC13, S. 254 ff.]. Das Kombinieren von mehreren Weights ist ebenfalls möglich [vgl. KC13, S. 254 ff.].

Tabelle 2.7 beinhaltet einen Ausschnitt von Slack Functions, welche von [KC13] vorgestellt wurden. Die Slack Functions können über die ebenfalls von [KC13] vorgestellten Weights aus Tabelle 2.8 gewichtet werden.

Slack Function	Beschreibung	Definition
$SF1$	Summe von freien Puffern	$\sum_{j=1}^{n} s_j$
$SF2$	Summe von binären Puffern	$\sum_{j=1}^{n} \alpha$ mit $\alpha = \begin{cases} 1 & \text{wenn } s_j > 0 \\ 0 & \text{sonst} \end{cases}$
$SF3$	Minimum zwischen freien Puffer und Bruchteil der Aktivitätsdauer d_j	$\sum_{j=1}^{n} \min(s_j, frac * d_j)$ mit $frac \in (0,1)$
$SF4$	Funktion zum Verringern des freien Puffers	$\sum_{i=1}^{s_j} e^{-i}$
...	...	...

Tabelle 2.7.: Definitionen von Slack Functions zur Robustheitsmessung

Quelle: In Anlehnung an [KC13, S. 254]

Weight	Beschreibung	Definition
$W1$	Anzahl der direkten Nachfolger	NDS_j
$W2$	Anzahl der benötigten Ressourcen	$\sum_{k=1}^{K} r_{i,k}$
$W3$	Kombination von $W1$ und $W2$	$NDS_j * \sum_{k=1}^{K} r_{i,k}$
$W4$	Anzahl der totalen Nachfolger	NS_j
...	...	...

Tabelle 2.8.: Definitionen von Weights für die Gewichtung der Slacks innerhalb der Slack Functions zur Robustsheitmessung

Quelle: In Anlehnung an [KC13, S. 255]

Durch das Auswählen unterschiedlicher Slack Functions und Weights können eine

Vielzahl von Robustheitsfunktionen hergeleitet werden. Eine Robustheitsfunktion Ω, welche die Slack Function $SF2$ mit dem Weight $W1$ vorsieht, würde gemäß [KC13] wie folgt definiert sein: $\Omega_{W1}^{SF2} = \sum_{j=1}^{n} \alpha * NDS_j$.

2.1.4.2. Reaktive Methoden

Unter reaktiven Methoden werden Verfahren verstanden, welche zum Zeitpunkt der Unsicherheit direkt reagieren, um so Verspätungen innerhalb der Projektdauer C_{max} zu minimieren. [vgl. BKF12, S. 404 f.].

Ein reaktiver Ansatz besteht darin, einen bestehenden Zeitplan zu einem Unsicherheitszeitpunkt zu reparieren. Hierbei werden zwischen proaktiven-reaktiven und prädiktiven-reaktiven Verfahren unterschieden. Diese Klassifizierung hängt von dem Baseline Schedule (zu dt. Basiszeitplan) ab, welcher initial festzulegen ist. Bei einem proaktiven-reaktiven Verfahren wird ein optimaler Zeitplan anhand der minimalen Projektdauer C_{max} ausgewählt, während beim prädiktiven-reaktiven Verfahren ein optimaler Zeitplan sowohl anhand der minimalen Projektdauer C_{max}, als auch an einem weiteren Kriterium, wie die maximale Robustheit Ω ausgewählt wird [vgl. BKF12, S. 404 f.].

Beim Reparieren gilt es zum Unsicherheitszeitpunkt neue Zeitpläne zu finden, welche zum einen gültig sind und zum anderen nah an dem Basiszeitplan liegen. Für das Finden von solchen Zeitplänen ist die Zielfunktion $\mathcal{C} = \sum_{i \in N} w_i |s'_i - s_i| + \sum_{i \in N} c_{im'_i}$ vorgesehen, welche die Rescheduling Kosten darstellen die es zu minimieren gilt. Hierbei stellen s_i die Startzeitpunkte der Aktivitäten des Basiszeitplans und s'_i die Startzeitpunkte für die Aktivitäten innerhalb des reparierenden Zeitplans dar. w_i sind Inflexibilitätsgewichte, welche die Intensität von Aktivitätsstartabweichungen bestimmen. Im zweiten Teil der Zielfunktionen werden die Moduswechselkosten $c_{im'_i}$ aller Aktivitäten aufaddiert. Diese betragen $c_{im'_i} = 0$, wenn der Modus für eine Aktivität zum Basiszeitplan nicht gewechselt wird. Die Inflexibilitätsgewichte w_i und die Moduswechselkosten $c_{im'_i}$ können im Vorfeld für einen Projektplan bestimmt werden. Mit diesen Werten ist es in der Praxis somit möglich, eher komplexere Änderungen dahingehend zu bestrafen, sodass diese für Folgezeitpläne weniger selektiert werden. [vgl. DDH08, S. 5 f.]

Eine weitere Kategorie von reaktiven Methoden stellen die dynamischen Scheduling Methoden dar. Diese basieren nicht auf einem Basiszeitplan und erstellen Zeitpläne zur Laufzeit mithilfe von Regeln. [BKF12, S. 404] Diese Masterthesis beschränkt sich im Rahmen der reaktiven Verfahren mit dem Reparieren von Zeitplänen anhand eines Basisplans.

2.2. Metaheuristiken

Optimierungsprobleme lassen sich im Grundlegenden in zwei Kategorien klassifizieren, nämlich in diskrete und kontinuierliche Optimierungsprobleme. Bei dem diskreten Fall existiert zur Lösungsfindung kein Wissen über einen exakten polynomialen Algorithmus und somit handelt es sich hierbei um ein NP-hartes Problem. Beim kontinuierlichen Fall existiert kein Wissen über einen Algorithmus zur Findung des globalen Optimums, welche die bestmögliche Lösung innerhalb eines endlichen Lösungsraums darstellt. [vgl. Sia16, S. 2]

Heuristiken können in diskreten Optimierungsproblemen eingesetzt werden, um passable Lösungen zu identifizieren, z. B. über Aktivititätsregeln beim RCPSP (vgl. Abschnitt 2.1.3.2). Diese Heuristiken sind dennoch kein Garant für das Finden vom globalen Optimum. Zudem sind Heuristiken für jeweils ein konkretes Problem entwickelt worden und können nicht auf andere Optimierungsprobleme angewandt werden [vgl. Sia16, S. 2].

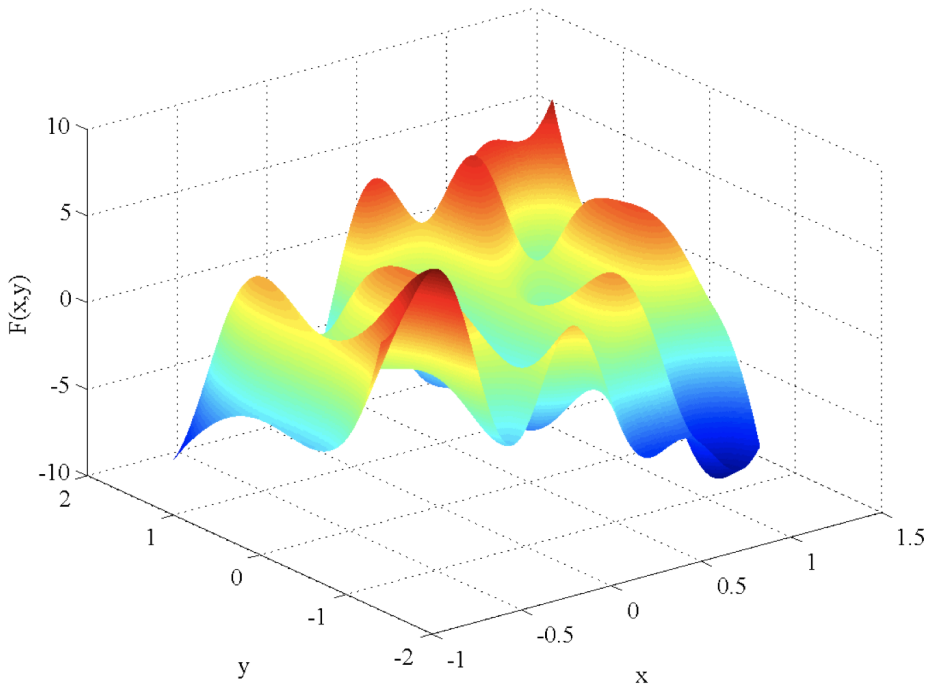


Abbildung 2.9.: Beispielhafte Darstellung eines kontinuierlichen Such- oder Optimierungsraums einer Zielfunktion $f(x, y)$ mit zwei Variablen x, y.

Quelle: [KKA16, S. 627]

Abbildung 2.9 stellt einen kontinuierlichen Suchraum einer beliebigen Funktion $f(x, y)$ mit zwei Variablen dar. Das Ziel zur Lösung eines Optimierungsproblems besteht darin, die Definitionen der Variablen zu bestimmen, für welche die Costs

(dt. Kosten) gemäß der Cost Function (dt. Kostenfunktion) am minimalsten bzw. maximalsten ist. In der Abbildung ist eine Vielzahl von lokalen Optima zu erkennen. Listing 2.2 beschreibt einen Local Search (LS)-Algorithmus, welcher anhand einer initialen Lösung (z. B. durch das Anwenden von Heuristiken generiert) über eine Anzahl an Iterationen oder einer Zeitvorgabe iterativ die nächstbeste Lösung innerhalb einer Nachbarschaft $N(s)$ auswählt. Das Problem hinter dieser Suchlogik besteht darin, dass dadurch ein lokales Optimum erreicht wird, welches nicht mehr über den Algorithmus verlassen werden kann. [vgl. MEPKQJ04, S. 3]

Listing 2.2: Local Search (LS)

```
1 Create an initial solution s inside the search space;
2 while stopping criteria not satisfied do
3     Select best solution s' ∈ N(s);
4     s ← s';
5 end
6 return the best solution;
```

Innerhalb eines hyperdimensionalen Lösungsraums, wie dies beim RCPSP der Fall ist, kann die Anwendung von der naiven lokalen Suche abhängig vom Szenario zu unbefriedigenden Ergebnissen führen. Die Ergebnisse können mehr oder weniger stark vom globalen Optimum abweichen.

Als Teilbereich der künstlichen Intelligenzen stellen Metaheuristiken die Werkzeuge zur intelligenten Suche von Lösungen innerhalb von Such- und Optimierungsproblemen dar. Das Ziel von Metaheuristiken gegenüber der lokalen Suche liegt darin, dass lokale Optima vermieden oder verlassen werden können, um so das globale Optimum eines Lösungsraums zu finden. [vgl. MEPKQJ04, S. 3]

Der Vorteil von Metaheuristiken gegenüber Heuristiken liegt darin, dass diese für alle Arten von diskreten Optimierungsproblemen generisch einsetzbar sind. Das Umgehen mit dem Phänomen der kombinatorischen Explosion innerhalb eines Optimierungsproblems, die Vermeidung der Nutzung von Gradientenberechnungen über die Zielfunktion, die Anlehnung an Naturwissenschaften orientierten Ansätzen, welche in der Physik, Biologie oder Ethologie vorzufinden sind, stellen weitere Vorteile und Merkmale von Metaheuristiken dar. Nachteile von Metaheuristiken sind jedoch die Bestimmung der Hyperparameter innerhalb der Algorithmen und die möglicherweise längere Berechnungsdauer. [vgl. MEPKQJ04, S. 2 f.]

Dieser Abschnitt befasst sich mit der Einführung von populären Metaheuristiken für diskrete Optimierungsprobleme. Zuerst wird die Tabu Search im Abschnitt 2.2.1

vorgestellt. Eine in der Metallurgie inspirierte Metaheuristik stellt Simulated Annealing (dt. simuliertes Abkühlen) dar, welche im Abschnitt 2.2.2 eingeführt wird. Evolutionäre Algorithmen, konkret die genetischen Algorithmen, stellen eine aus der Biologie orientierte Metaheuristik dar. Diese werden im Abschnitt 2.2.3 behandelt.

2.2.1. Tabu Search

Die Tabu Search (zu dt. Tabu-Suche) wurde von Fred Glover im Jahre 1986 vorgeschlagen [vgl. GP19, S. 37] und gilt als eine Erweiterung der Local Search (LS), welche bereits im übergeordneten Abschnitt eingeführt wurde. Der Vorteil der Tabu Search (TS) gegenüber der LS ist der Umgang mit lokalen Optima über das Nutzen einer Tabu-Liste. Hierbei werden Lösungen innerhalb einer Nachbarschaft ausgeschlossen, die sich bereits in der Tabu-Liste befinden. Die lokale Suche samt Einhaltung einer Tabu-Liste als Kurzzeitgedächtnis stellt die Basisversion der TS dar. [vgl. GP19, S. 40]

Beim Erreichen eines lokalen Optimums findet innerhalb der LS in jeder Iteration ein repetitiver Wechsel zweier Lösungen statt. Durch diese Wechselschleife bis zur Terminierung des Algorithmus wird das lokale Optimum nicht mehr verlassen. [vgl. GP19, S. 40]

Listing 2.3: Tabu Search (Quelle: [vgl. GP19, S. 42 ff.])

```
1 Create an initial solution s inside the search space;
2 Init tabu list TL ← ∅;
3 while stopping criteria not satisfied do
4     Select best solution s' ∈ N(s) \ TL;
5     s ← s';
6     Add s' to the tabu list TL;
7     If tabu list TL is full, remove oldest entry;
8 end
9 return the best solution;
```

Listing 2.3 beschreibt den Algorithmus der Tabu-Suche. Eine Tabu-Liste TS stellt das Kurzzeitgedächtnis dar, welches nur eine feste Anzahl an Einträgen beinhalten kann und im initialen Zustand zunächst leer ist. Die Größe der Tabu-Liste $|TS]$ stellt ein Hyperparameter dar, welcher somit bei der Implementierung zu definieren ist. In jeder Iteration wird die ausgewählte Lösung, welche der besten Lösung einer Nachbarschaft entspricht, in die Tabu-Liste an erster Stelle hinzugefügt. Bestehende Einträge werden folglich um eine Stelle in der Liste weiterpropagiert. Einträge aus der Tabu-Liste dürfen nicht mehr aus einer Nachbarschaft $N(s)$ ausgewählt werden. Sofern die Tabu-Liste voll ist, wird das älteste Element entfernt und darf somit wie-

der ausgewählt werden. Durch das Ausschließen der Listeneinträge ist ein repetitiver Wechsel zweier Lösungen, wie es bei der LS der Fall ist, je nach Größe der Tabu-Liste vermeidbar. Zudem wird der Suchraum um einen größeren Bereich erkundet und lokale Optima können verlassen werden. [vgl. GP19, S. 42 ff.]

Sowohl in der lokalen Suche als auch in der Tabu-Suche ist eine Nachbarschaftsfunktion vorgesehen. Die Nachbarschaftsfunktion $N(s)$ stellt eine Submenge von Lösungen eines Suchraums dar. Diese beinhaltet Lösungen, welche von einer Ausgangslösung s mit einem Move m (dt. Schritt) erreichbar sind. Ein Move m wird als eine charakterisierte Modifikation an einer Lösung bezeichnet. [vgl. Sia16, S. 56]

Beispielhaft visualisiert Abbildung 2.10 den Lösungsraum einschließlich der Nachbarschaftsbeziehungen von allen Permutationen von vier Elementen. Die Nachbarschaftsbeziehungen unterscheiden sich je nach Optimierungsproblem und müssen folglich für jedes Problem gesondert selektiert werden [vgl. Sia16, S. 57 f.]. Gemäß des Beispiels in der Abbildung wäre die Nachbarschaft für die Lösung 1234 über $N(1234) = \{1324, 1432, 2134\}$ definiert, da nur ein Move notwendig ist, um diese Lösungen zu erreichen.

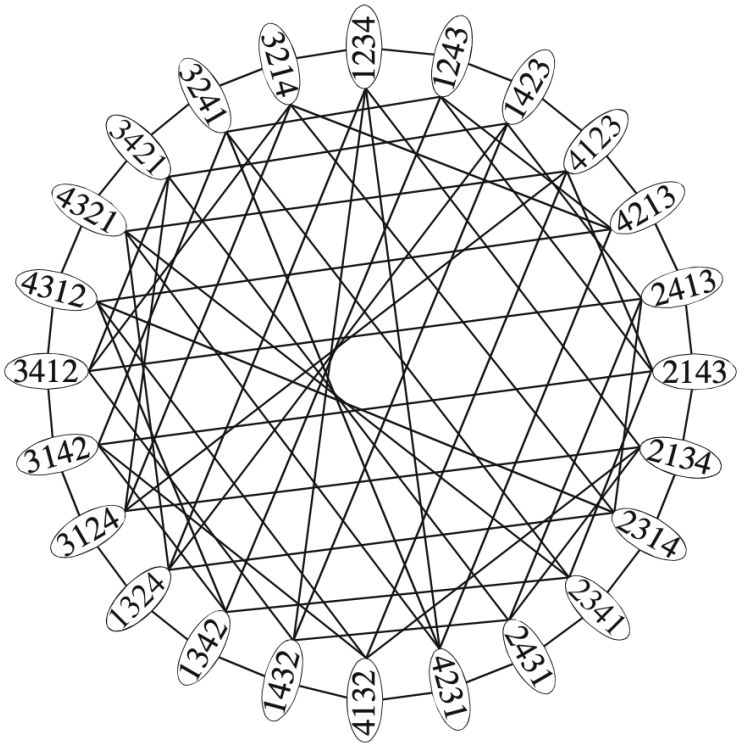


Abbildung 2.10.: Nachbarschaftsbeziehungen einer Menge von Permutationen aus vier Elementen in Knotenform

Quelle: [Sia16, S. 57]

Bei der Tabu-Suche stehen wie auch bei anderen Metaheuristiken, unterschiedliche Möglichkeiten zur Terminierung der Algorithmen zur Verfügung. Ein Algorithmus kann nach einer festen Anzahl an Iterationen oder CPU-Zeit beendet werden. Ein Algorithmus kann zudem auch nach einer bestimmten Anzahl an Iterationen, wo keine Verbesserung der Zielfunktion vorzufinden ist, terminiert werden. Eine weitere Möglichkeit ist, dass ein Algorithmus solange durchiteriert wird, bis ein definierter Schwellwert erreicht wurde. [vgl. GP19, S. 44]

2.2.2. Simulated Annealing

Eine an der Physik, nämlich an der Metallurgie, orientierte Metaheuristik stellt Simulated Annealing (SA) (dt. simuliertes Abkühlen) dar und wurde von den 3 IBM Wissenschaftlern Kirkpatrick, Gelatt und Vecchi im Jahre 1992 vorgeschlagen und im Jahr 1993 veröffentlicht. [vgl. Sia16, S. 19]

Für einen gegebenen Körper gilt es, nachdem dem Körper eine sehr hohe Temperatur zugeführt wurde, einen energetisch günstigen Zustand zu erreichen. Durch das Erhöhen der Temperatur zu einem sehr hohen Punkt wird die Struktur des Körpers zunächst geschmolzen. Der Körper befindet sich somit in einer flüssigen Phase, in welcher die Partikel des Körpers zufällig verteilt sind. Der Körper wird über das Abkühlen gemäß eines besonderen Temperaturschemas wieder in eine stabile Phase zurückgeführt und erreicht somit einen energetisch günstigen Zustand. Sowohl die initiale Temperatur als auch die Kühlungszeit müssen eine entsprechende Höhe aufweisen, da ansonsten ein metastabiler Zustand erreicht wird, in welcher die Energie nicht minimal ist. Der Prozess wird Härtung genannt, wenn die Temperaturabnahme nicht stetig ist und durch eine abrupte Abkühlung beeinflusst wird. Die Funktionsweisen und die Unterschiede zwischen dem Härtungs- und dem Abkühlungsprozess lassen sich über die Abbildung 2.11 visualisieren. [vgl. GP19, S. 2 f.]

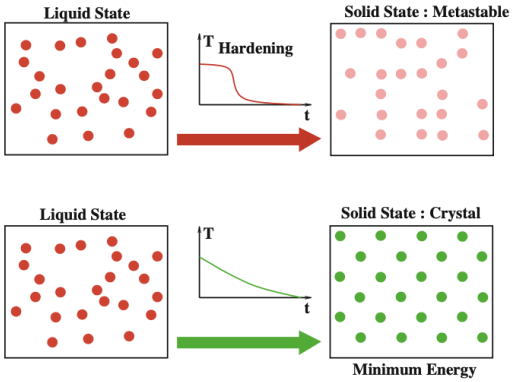


Abbildung 2.11.: Funktionsweise der Abkühlungs- und Härtungsprozesse

Quelle: [GP19, S. 3]

Zwischen einem Optimierungsproblem und dem physikalischen System lassen sich Analogien ziehen. In der Physik wird die freie Energie betrachtet, bei dem Optimierungsproblem die Zielfunktion. Die Positionierung der Partikel stellt die Analogie zu den Parametern eines Problems dar. Das Ziel innerhalb eines physikalischen System liegt darin, dass ein energiearmer Zustand erreicht wird. Dies trifft ebenfalls bei Optimierungsproblemen zu, bei welchen Konfigurationen gesucht werden, welche als „gut“ klassifiziert werden können oder im besten Fall am optimalsten sind. [vgl. Sia16, S. 20]

Listing 2.4: Simulated Annealing (Quelle: [vgl. HASB17, S. 714])

```
1  Create an initial solution s inside the search space;
2  Select initial temperatur T_0;
3  Select rate of temperature decrease α ∈ [0.80, 0.99];
4  T ← T_0;
5  while stopping criteria not satisfied do
6      Select random solution s' ∈ N(s);
7      if f(s') < f(s)
8          s ← s';
9      else
10         Δ ← f(s') < f(s);
11         if random(0, 1) < exp(−Δ/T)
12             s ← s';
13     T ← T * α;
14 end
15 return the best solution;
```

Innerhalb des Algorithmus aus Listing 2.4 wird zunächst eine initiale Temperatur T_0 ausgewählt. Die Senkung anhand eines Temperaturschemas wird über eine Temperatursenkungsrate $\alpha \in [0.80, 0.99]$ realisiert. Wie auch bei der LS wird eine initiale Lösung s benötigt, welche zufällig oder über Heuristiken erzeugt werden können. Die initiale Lösung und Temperatur entsprechen dem flüssigen Zustand im Abkühlungsprozess. Nach jeder Iteration innerhalb des Algorithmus wird die Temperatur mit der Temperatursenkungsrate multipliziert $T \leftarrow T * \alpha$, was nach jeder Iteration zu einer Senkung der Temperatur führt. Um die Analogie zur Verfestigung zu realisieren, nutzt der Algorithmus die Nachbarschaftsfunktion, um so eine zufällige Lösung s' auszuwählen. Über die Nutzung einer Akzeptanzfunktion gemäß Formel 2.12 [vgl. GP19, S. 6] und das Generieren einer zufälligen Zahl $r \in [0, 1]$ wird die Akzeptanz der Lösung s' bestimmt. Sofern $r \leq P(s, s')$ gilt, wird die aktuelle Lösung überschrieben $s \leftarrow s'$. Das Annehmen von verschlechternden Lösungen im Algorithmus trägt maßgeblich zur Erkundung des Suchraums bei. Die Akzeptanz von schlechteren Lösungen sinkt über die Iterationen, da die ebenfalls sinkende Tem-

peratur maßgebend zum Akzeptanzwert ist. Eine sinkende Temperatur impliziert, dass die Partikel innerhalb eines physikalischen Systems sich weniger stark bewegen. Gänzlich werden bei einer Temperatur $T \approx 0$ schlechtere Lösungen abgelehnt, was dazu führt, dass der Algorithmus lediglich als lokale Suche agiert. Es werden dann nur Lösungen angenommen, wenn die selektierte Lösung aus der Nachbarschaft besser als die aktuelle ist. [vgl. GP19, S. 5 f.] [vgl. Sia16, S. 21 f.]

$$P(s, s') = \begin{cases} 1 & \text{wenn } f(s') < f(s) \\ e^{\left(\frac{f(s) - f(s')}{T}\right)} & \text{sonst} \end{cases} \tag{2.12}$$

Wie bei der TS stehen beim SA unterschiedliche Stopkriterien zur Verfügung. Der Algorithmus von [GP19, S. 6 f.] iteriert solange, bis die Temperatur bei $T \approx 0$ liegt. Der Algorithmus von [HASB17, S. 714] nutzt zusätzlich eine feste Anzahl an Iterationen, die durchlaufen werden müssen. Alternativ können die aus dem Abschnitt 2.2.1 für die TS vorgestellten Kriterien verwendet werden. Ein weitere Besonderheit stellt die Verwendung einer Boltzmann Konstante k_B dar, welche zur Gewichtung der Temperatur in der Annahmefunktion beiträgt [vgl. Sia16, S. 21]. Durch die Verwendung einer Boltzmann Konstante k_B lässt sich die Relevanz der Temperatur bei der Akzeptanzfunktion verändern.

2.2.3. Evolutionäre Algorithmen

Evolutionäre Algorithmen stellen im Bereich der Metaheuristiken und in künstlichen Intelligenzen einen großen Forschungsbereich dar. Die Basis der evolutionären Algorithmen führt auf Charles R. Darwin zurück, welcher als Vater der Evolutionstheorie bekannt ist. Lebewesen, die sich innerhalb ihrer Umgebung am besten anpassen können setzten sich durch und übergeben ihren Nachfahren nützliche überlebenswichtige Eigenschaften. [vgl. Sia16, S. 115]

Der Oberbegriff „evolutionäre Algorithmen“ fasst die Basis der Evolutionstheorie auf und wird genutzt, um Probleme in den Ingenieurswissenschaften zu lösen. Evolutionäre Strategien wurden in den 60er Jahren von Schwefel und Rechenberg vorgestellt und nutzen die Genetik, um Optimierungsprobleme mit kontinuierlichen Variablen zu lösen. Mitte der 60er wurde von Fogel et al. die evolutionäre Programmierung vorgestellt. Die genetischen Algorithmen, vorgestellt in 1975 von Holland, als Forschungsfeld populär geworden in 1989 von Goldberg, tragen dazu bei, die Funktionsweisen von selbstanpassenden Systemen zu verstehen. Genetische Algorithmen werden zudem als Metaheuristik genutzt, um Lösungen für Optimierungs-

probleme zu finden. [vgl. Sia16, S. 116]
Diese Arbeit beschränkt sich somit auf die GAs, eine Untermenge der evolutionären Algorithmen.

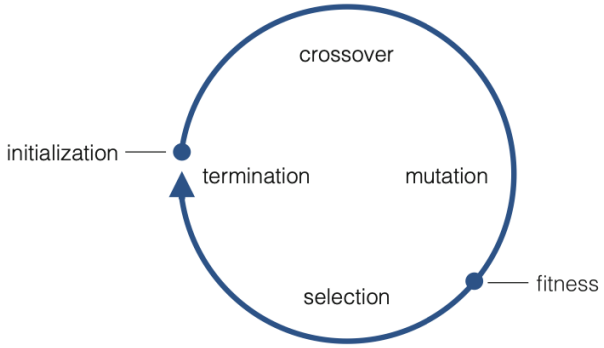


Abbildung 2.12.: Kreislauf eines Genetic Algorithm (GA)

Quelle: [Kra17, S. 5]

Abbildung 2.12 zeigt den Kreislauf eines genetischen Algorithmus, basierend auf der natürlichen Evolution. Zu Beginn wird eine Population generiert, was zufällig oder manuell (z. B. über heuristische Regeln) geschehen kann. Über eine Crossover-Operation werden zwei oder mehrere Eltern rekombiniert, um so ein oder mehrere Nachfahren zu erzeugen. Zudem sind die Nachfahren nach dem Crossover gemäß einer Mutations-Operation mutiert. In der Selektionsphase werden entsprechend einer Regel die besten Ergebnisse gemäß ihres Fitnesswertes ausgewählt. Der Algorithmus läuft solange durch, bis das beliebige Terminierungskriterium erfüllt wurde. [vgl. Kra17, S. 5]

Listing 2.5: Genetic Algorithm (Quelle: [vgl. Sia16, S. 119])

```
1  Select size of parents μ;
2  Select size of offsprings λ;
3  init population P with μ individuals;
4  fitness evaluation ∀i ∈ P : f(i) of population P;
5  while stopping criteria not satisfied do
6      for i in range(0, λ) do
7          select parents κ from population;
8          x ← crossover κ;
9          x ← mutate x;
10         fitness evaluation f(x);
11     end
12     selection of μ individuals for next generation population P
13 end
14 return the best individual;
```

Listing 2.5 stellt den Pseudocode des Algorithmus dar. μ steht für die Anzahl der Eltern, λ für die der Nachkommen [vgl. Sia16, S. 119]. Der aufgeführte Algorithmus ist generisch gehalten, da für die Crossover-, Mutations- und Selektionsoperationen [vgl. Sia16, S. 118] verschiedene Strategien je nach Problemart existieren. Außerdem für das Problem angeschnitten ist die Repräsentation eines Individuums. Diese kann beispielsweise eine binäre [vgl. Sia16, S. 137], reale [vgl. Sia16, S. 140] oder beliebige diskrete [vgl. Sia16, S. 149] Struktur aufweisen.

Bei der Crossover-Operation findet eine Rekombination zweier oder mehrerer (Eltern-)Individuuen μ_1, μ_2 statt. Eine einfache Crossover-Operation für kontinuierliche Variablen ist die Verwendung eines Arithmetic Crossover. Hierbei wird das arithmetische Mittel aus den Eltern berechnet, welches der Nachfahre darstellt. Bei zwei Eltern mit $\mu_1 = (1, 2, 3)$ und $\mu_2 = (5, 4, 3)$ wäre der Nachfahre $\lambda_1 = (3, 3, 3)$. [vgl. Kra17, S. 13]

Im Hinblick auf die Masterthesis stehen beim GA die diskreten Strukturen im Vordergrund. Eine diskrete Struktur bei Individuen könnten Reihenfolgen darstellen, in welcher Sequenzen nur einmalig vorhanden sein können[vgl. Sia16, S. 151], oder ggf. mit Vorgängeraktivitäten verknüpft sind, wie dies bei Aktivitätslisten beim (M)RCPSP der Fall ist (vgl. Abschnitt 2.1.3.2).

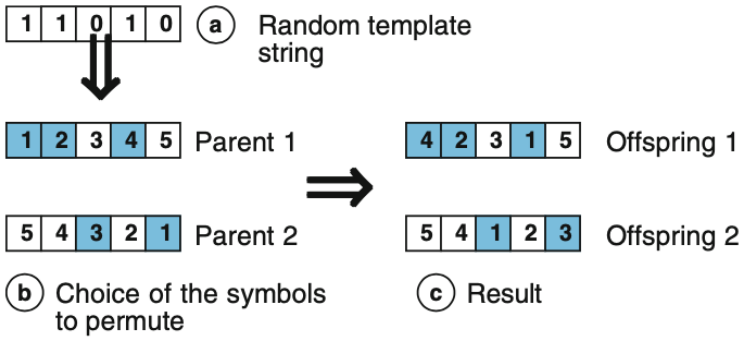


Abbildung 2.13.: Uniform Order-Based Crossover

Quelle: [Sia16, S. 151]

Abbildung 2.13 stellt einen Uniform Order-Based Crossover dar, welcher sich in drei Phasen einteilen lässt und sich auf zwei Eltern-Individuen anwenden lässt. Zunächst wird ein zufällig generiertes binäres Template verwendet, welches die Positionen der Sequenzen der Eltern die zur Rekombination verwendet wird, vormerkt. Im Beispiel sind bei Parent 1 die Sequenzen 1, 2 und 4, bei Parent 2 die Sequenzen 3 und 1 vorgemerkt. Das Vormerken gibt an, dass die Elemente gemäß der Reihenfolge

des anderen Elternteils permutiert werden können. [vgl. Sia16, S. 151]

Bei einer Mutation-Operation werden kleine, zufällige Änderungen an einem (Nachkommen-)Individuum λ_1 vorgenommen. Die Mutation dient als Möglichkeit, den Suchraum weiter außerhalb der Population P zu erkunden. Bei kontinuierlichen Variablen stellt die Gaussian Mutation eine etablierte Operation dar. Hierbei wird eine zufällige reale Zahl von der Normalfunktion $\mathcal{N}(0,1)$ generiert. Sei σ die Mutationsrate, dann wäre ein Nachkommen λ_1 über $\lambda_1' = \lambda_1 \cdot \mathcal{N}(0,1)$ mutiert. [vgl. Kra17, S. 13 f.]

Die Mutation durch Austausche-Operation stellt eine mögliche Variante für diskrete Variablen mit Reihenfolgenbeziehungen dar. Diese wählt zwei Sequenzen innerhalb eines Individuums λ_1 aus und vertauscht diese miteinander, wenn die Beschränkungen gemäß des Problems eingehalten werden können. [vgl. Sia16, S. 154]

Eine weitere Komponente von genetischen Algorithmen stellt die Fitness eines Individuums dar. Die Fitness eines Individuums sagt aus, wie gut die Qualität hinter dem Individuum ist [Kra17, S. 16 f.]. Dies kann im Fall von Optimierungsproblemen die Kostenfunktion (z. B. die Projektdauer beim RCPSP) sein. Zudem müssen Aspekte, wie Multi-Objectives oder Bestrafen durch Verletzungen von Beschränkungen über die Fitness-Funktion abgedeckt werden [vgl. Kra17, S. 16 f.].

Die Selektion stellt im generischen GA die letzte Operation einer Iteration bzw. der Generation dar und wählt die Individuen für die Folge-Population aus. Für die Auswahl der Individuen zur nächsten Generation stehen unterschiedliche Varianten zur Verfügung. Eine Variante stellt ein (μ, λ)-GA dar. Diese entspricht der Komma-Selektion, welche die μ besten Lösungen aus den λ erzeugten Nachfahren der aktuellen Generation selektiert. $(\mu + \lambda)$-GA verwendet die Plus-Selektion, d.h. die μ besten Lösungen sowohl aus den λ erzeugten Nachfahren als auch die Eltern μ der aktuellen Generation werden selektiert. [vgl. Kra17, S. 16 f.].

Die Terminierung eines GA geschieht z. B. über das Durchlaufen einer festen Anzahl an Generationen, der CPU-Zeit, das Erreichen eines vorgegebenen Fitnesswertes oder nach Feststellung einer Sättigung et cetera. [vgl. Kra17, S. 17 f.].

2.3. Stand der Forschung

Der Einsatz von Metaheuristiken für das (M)RCPSP stellt keineswegs einen unerforschter Bereich dar. Bereits 1998 verglichen [KH98] die Leistungen der Metaheu-

ristiken für das Finden von optimalen Zeitplänen im Bezug auf die Projektdauer. Hierbei wurden unterschiedlich implementierte Metaheuristiken wie die Tabu Search, Simulated Annealing und Genetic Algorithm von Wissenschaftlern für das RCPSP quantitativ verglichen [vgl. KH98, S. 17 f.].

Für die Multi-Mode Problemerweiterung wurden ebenfalls bereits eine Vielzahl an Metaheuristiken entworfen. Allein für die genetischen Algorithmen existiert eine Vielzahl von wissenschaftlichen Veröffentlichungen, wie der im Jahre 2003 publizierte Artikel von [AMR03]. Weitere Artikel stellen beispielsweise die im Jahr 2013 veröffentlichte Arbeit von [LG13] oder 2015 von [SAA15] dar. Diese Arbeiten beziehen sich alle auf das MRCPSP, aber unterscheiden sich unter anderem stark ihren Repräsentationsformen, Crossover-, Mutations- und Selektionsoperatoren oder die Wahl der Hyperparameter. Abseits von den genetischen Algorithmen wurde unter anderem 2001 von [JMR+01] ein Simulated Annealing-Algorithmus für das MRCPSP implementiert. Innerhalb eines Papers zum reaktiven Scheduling wurde 2011 von [DDH11] ein Tabu Search-Algorithmus umgesetzt.

Das Thema der Unsicherheiten und der prädiktive Umgang wurde 2005 von [AFH05] in einem Artikel behandelt. In Kombination mit einer multi-objective Tabu Search gilt es die Projektdauer zu minimieren und zugleich die Robustheit zu maximieren. Durch die Maximierung der Robustheit können Pläne im Vornherein gefunden und die Projektverspätungen durch eine erhöhte Puffergröße minimiert werden [vgl. AFH05, S. 177 f.].

Abseits der Maximierung der Summe aller Aktivitätspuffer existiert eine Vielzahl an Robustheitsfunktionen. 2013 wurden im Artikel von [KC13] bestehende und neue Robustheitsmessungen für das RCPSP verglichen.

Die Verspätungen durch Unsicherheiten können zudem reaktiv gemindert werden. 2011 nutzte [DDH11] als reaktives Verfahren für das MRCPSP unter anderem baumbasierte Suchtechniken, um so die Reschedule Costs bei den Unsicherheitszeitpunkten zu minimieren [vgl. DDH11, S. 66].

2012 wurde in einer Erhebung von [BKF12] sowohl proaktive, prädiktive als auch für die reaktive Verfahren sowohl für das RCPSP als auch die stochastische Version SRCPSP aufgeführt. Im Fazit der Erhebung wurde anhand Daten verschiedener Publikationen hervorgehoben, dass im Bezug zum Basisproblem proaktive Verfahren besser für quantifizierbare Unsicherheiten und reaktive Verfahren geeigneter für weitaus größere Unsicherheiten sind.

Diese Masterarbeit baut auf den Ausblick von [BKF12, S. 405 f.] auf und betrachtet Unsicherheiten für das MRCPSP, was eine generalisierte Variante des RCPSP darstellt. Die zitierte Erhebung baut auf quantitative Daten einzelner Publikationen auf, welche unterschiedliche Unsicherheitsszenarien und Verfahren zur Lösung des MRCPSP aufweisen. Es gilt somit festzustellen, wie sich die Verfahren aus den pro-, prä- und reaktiven Methoden quantitativ auf eine gemeinsame Basis von Unsicherheitsszenarien und Metaheuristiken auswirken. Zudem kann evaluiert werden, wie sich die Leistung der unterschiedlichen Metaheuristiken nach Anwenden der Unsicherheitsszenarien auf die Projektdauer bei den einzelnen Verfahren auswirken. Die aufgeführten Themen liefern zum Zeitpunkt der Thesis noch Forschungsbedarf, was die Existenz der Forschungsfrage und dessen Unterfragen aus Abschnitt 1.2 begründen soll.

Kapitel 3

Konzept des MRCPSP-Frameworks

Dieses Kapitel befasst sich mit der Konzeption eines MRCPSP-Frameworks, welches zur Beantwortung der Forschungsfrage und dessen Unterfragen maßgeblich ist. Hierfür wird im Abschnitt 3.1 zunächst ein Überblick der Anforderungen gegeben, welche zu implementieren sind. Basierend auf den Anforderungen werden im selben Abschnitt die Teilaspekte des Frameworks hergeleitet. Konzeptionsentscheidungen, wie die Auswahl der Verfahren für den Umgang von Unsicherheitsentscheidungen (Abschnitt 3.2) und die Auswahl der Metaheuristiken (Abschnitt 3.3) werden ebenfalls im Kapitel behandelt. Um Vergleiche auf eine gemeinsame Datenbasis zu realisieren, befasst sich Abschnitt 3.4 mit der Auswahl des Benchmarkdatensatzes. Die Konzeption von Unsicherheitsszenarien wird im Abschnitt 3.5 behandelt.

3.1. Konzeptüberblick

Um die Auswirkungen der prädiktiven und reaktiven Ansätze für das MRCPSP auf Basis von metaheuristischen Algorithmen bei der Erstellung von Zeitplänen in Bezug auf die Projektdauer bei Verspätungen untersuchen zu können, ist eine Implementierung einer Simulationsumgebung unabdingbar. Diese Umgebung wird im Rahmen der Arbeit als MRCPSP Framework bezeichnet.

Diese Umgebung muss zunächst in der Lage sein, Zeitpläne für das MRCPSP anhand von Aktivitäts- und Moduslisten (vgl. Abschnitt 2.1.3.2 und Abschnitt 2.1.3.3) erstellen zu können. Des Weiteren gilt es mittels Metaheuristiken (vgl. Abschnitt 2.2) gute Zeitpläne im Bezug auf die minimale Projektdauer des MRCPSP zu finden. Dennoch bauen prädiktive Verfahren auf eine weitere Metrik auf, nämlich der Robustheit (vgl. Abschnitt 2.1.4.1). Somit müssen Metaheuristiken in der Lage sein, Pläne mit der minimalsten Projektdauer zu finden, welche die maximalste Robustheit aufweisen. Das Problem des Findens von Lösungen mit mehreren Zielfunktionen bezeichnet sich als Multi-Objective [vgl. ASA06, S. 146 f.]. Metaheuristiken müssen somit in der Lage sein, Zeitpläne anhand mehrerer Zielfunktionen auswählen zu

können. Um zunächst die Performanz der Metaheuristiken unabhängig der Unsicherheitsszenarien vergleichen zu können, muss eine Experimentumgebung geschaffen werden. Diese vergleicht quantitativ die implementierten Metaheuristiken miteinander.

Des Weiteren muss eine weitere Experimentumgebung für die Unsicherheitsszenarien realisiert werden. Zunächst gilt es die Unsicherheitsszenarien zu definieren, mit welchen die einzelnen Verfahren innerhalb des Experimentes konfrontiert werden müssen. Es müssen zudem die pro-, prä- und reaktiven Verfahren ausgewählt werden.

Die Experimente gilt es auf Benchmark-Instanzen anzuwenden. Hierfür müssen diese Instanzen zunächst ausgewählt und innerhalb des Frameworks hereingeladen werden können.

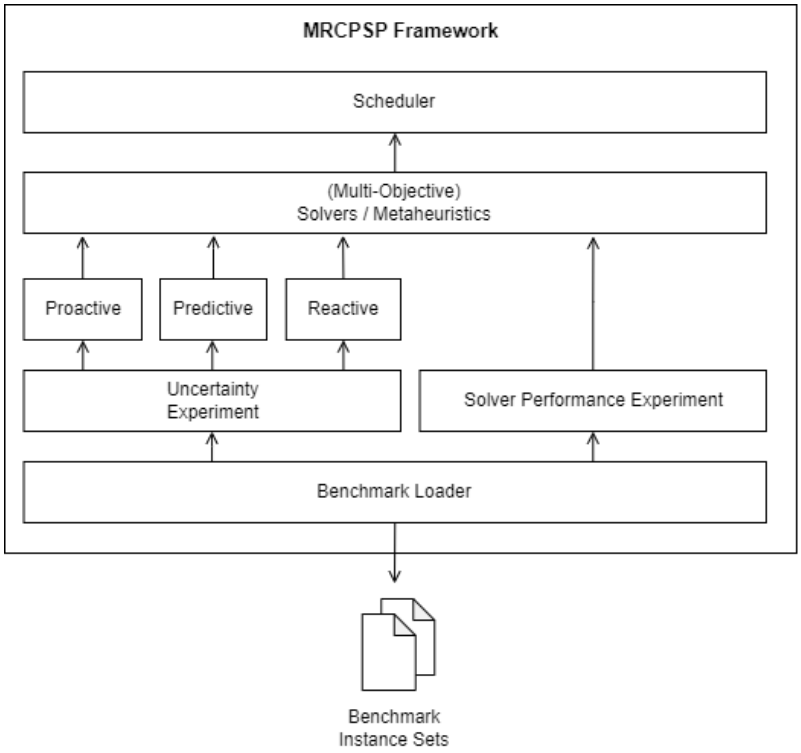


Abbildung 3.1.: Statischer Konzeptüberblick des MRPCSP-Framework

Quelle: Eigene Darstellung

Die definierten Anforderungen des MRCPSP Frameworks lassen sich über die Abbildung 3.1 visualisieren. Um das MRPCSP Framework zu realisieren wird als Programmiersprache Java ausgewählt. Unterstützend dazu wird das Spring Framework eingesetzt.

Bei der populären Programmiersprache Java stehen objektorientierte Programmierung, einfache Wartbarkeit komplexer Software und die Plattformunabhängigkeit durch die Java Virtual Environment im Vordergrund. C bietet als hardwarenahe Sprache dennoch den Vorteil der hohen Geschwindigkeit gegenüber Java. Der Nachteil von manueller Adressverwaltung und somit einer schwierigen Wartbarkeit komplexer Software macht den Vorteil jedoch wett. [vgl. Zan19]

Ein für Java entwickeltes leichtgewichtiges Open Source-Framework stellt Spring dar. Dieses erweitert Java mit dem Dependency Injection-Pattern, was unter anderem leicht entkoppelte Systemkomponenten ermöglicht. Insbesondere für Unternehmen findet Spring in Webanwendungen eine hohe Beliebtheit. Im Rahmen der Masterarbeit wird Spring für die Wartbarkeit und die Auswechselbarkeit von bestimmten Komponenten genutzt, um so den repetitiven Code zu minimieren. [vgl. Aug19]

Insgesamt lässt sich anhand des statischen Konzeptes für das MRCPSP-Framework ein Ablaufplan zur Beantwortung der Forschungsfrage herleiten. Dieser ist in Abbildung 3.2 aufgezeigt. An vielen Aspekten, wie der Selektion und Konzeption der Lösungs-verfahren, der Auswahl und Integrierung der Benchmarks oder der Generierung der Unsicherheitsszenarien kann unabhängig voneinander entwickelt werden. Erst die Experimente nutzen die verschiedenen Teilaspekte, um so die Güte der einzelnen Methoden und Verfahren quantitativ zu messen. Bei einem Experiment werden die Lösungsverfahren für jede Benchmark-Instanz durchlaufen. Zudem werden bei den Unsicherheitsexperimenten die generierten Unsicherheitsszenarien angewendet. Die Ergebnisse, wie die Makespan-Werte der Basis- und Verspätungszeitpläne werden innerhalb einer CSV-Datei gesammelt. Diese gilt es im Anschluss gemäß der Forschungsfrage zu evaluieren und werden detaillierter im Abschnitt 4.4 beschrieben.

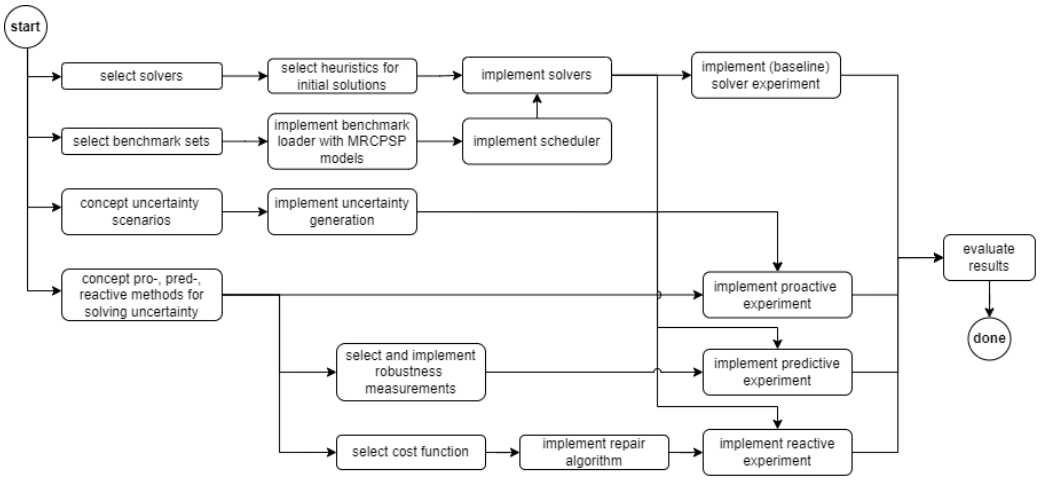


Abbildung 3.2.: Ablaufplan zur Beantwortung der Forschungsfrage

Quelle: Eigene Darstellung

3.2. Auswahl der Verfahren zur Lösung von Unsicherheiten

Bereits im Abschnitt 2.1.4 wurden Verfahren zur Lösung von Unsicherheiten vorgestellt. Aufgrund der zeitlichen Restriktionen dieser Arbeit werden jeweils ein konkretes Verfahren aus den Kategorien der pro-, prä-, und reaktiven Methoden verglichen. Ziel der Verfahren besteht darin, die verursachten Verspätungen der Unsicherheiten so gering wie möglich zu halten. Folgende Verfahren stehen repräsentativ für ihre jeweiligen Kategorien:

Proaktiv Bei proaktiven Verfahren werden die Unsicherheiten ignoriert. Zeitpläne werden somit nur nach der minimalen Makespan C_{max} selektiert. Dieses Verfahren dient im Rahmen der Masterarbeit als ein Referenzwert. Verfahren, die Unsicherheiten berücksichtigen, sollten bessere Ergebnisse als die proaktive Herangehensweise liefern.

Prädiktiv Als prädiktive Herangehensweise wird neben der Minimierung der Makespan C_{max} die Robustheit Ω von Zeitplänen maximiert. Es existieren für das RCPSP eine Vielzahl an Robustheitsmessungsfunktionen (vgl. Abschnitt 2.1.4.1). Innerhalb eines Experimentes werden unterschiedliche Robustheitsmessungen verglichen, um so aus den unterschiedlichen Robustheitsmessungsfunktion die bestmögliche Variante für den direkten Vergleich der anderen Verfahren auszuwählen.

Reaktiv Bei den reaktiven Verfahren wird zum Zeitpunkt der Unsicherheit reagiert. Diese Arbeit befasst sich im Rahmen der reaktiven Verfahren mit der

Reparatur von Zeitplänen zum Unsicherheitszeitpunkt. Die Kostenfunktion $\mathcal{C}$ wird über eine gesonderte Implementierung des Tabu-Search-Algorithmus minimiert, um so ähnliche Zeitpläne zum Basiszeitplan zu erhalten. In Anlehnung an die Arbeit von [DDH08] werden die Inflexiblilitätsgewichte über eine Binomialverteilung und Moduswechselkosten über eine Gleichverteilung zufällig generiert, wobei die Dummy-Endaktivität mit einem Gewicht von $w_n = 10 * |E(\mathcal{B}(n,p))|$ versehen ist. Der Erwartungswert der Binomialverteilung wird über einen Koeffizienten multipliziert, um so Verspätungen durch Unsicherheitsszenarien stärker zu bestrafen.

3.3. Auswahl der metaheuristischen Algorithmen

Im Abschnitt 2.2 wurden Metaheuristiken für Optimierungsprobleme im Generellen vorgestellt. Für das (M)RCPSP wurde bereits eine Vielzahl an Metaheuristiken konzipiert und miteinander verglichen [vgl. KH98, S. 16 ff.]. Wie im vorherigen Abschnitt 3.2 beschrieben, kann in dieser Arbeit aufgrund zeitlicher Restriktionen nur eine Auswahl an Metaheuristiken implementiert und verglichen werden.

Die implementierten Metaheuristken müssen in der Lage sein, mehreren Zielfunktionen herrzuwerden. Bei dem prädiktiven Verfahren soll neben der Minimierung der Makespan C_{max} die Robustheit Ω maximiert werden. Viele Multi-Objective Metaheuristiken nutzen daher Pareto-Prinzipien, um optimale Fronten zwischen mehreren Zielfunktionen zu definieren. Abbildung 3.3 zeigt eine beispielhafte Pareto-Front zweier Zielfunktionen auf. Während es bspw. f_1 zu maximieren gilt, soll f_2 minimiert werden. Dreiecke und Kreuze innerhalb der Grafik stellen schlechtere Kompromisse zweier Funktionen dar, da diese von anderen Lösungen, nämlich den Kreisen dominiert werden. [vgl. TBNA12, S. 285 ff.]

Im Rahmen der Arbeit werden die Zielfunktionen der Metaheuristiken strikt priorisiert. Die Makespan C_{max} gilt als die primäre Zielfunktion, welches bei einem direkten Vergleich von mehreren Lösungen zunächst die bessere Lösung gemäß der minimalen Makespan selektiert. Die sekundäre Zielfunktion, welche die Robustheit Ω eines Plans darstellt kommt nur in Betracht, wenn mehrere Lösungen mit identischer Makespan verglichen werden. Hierbei wird die Lösung ausgewählt, welche die maximale Robustheit aufweist.

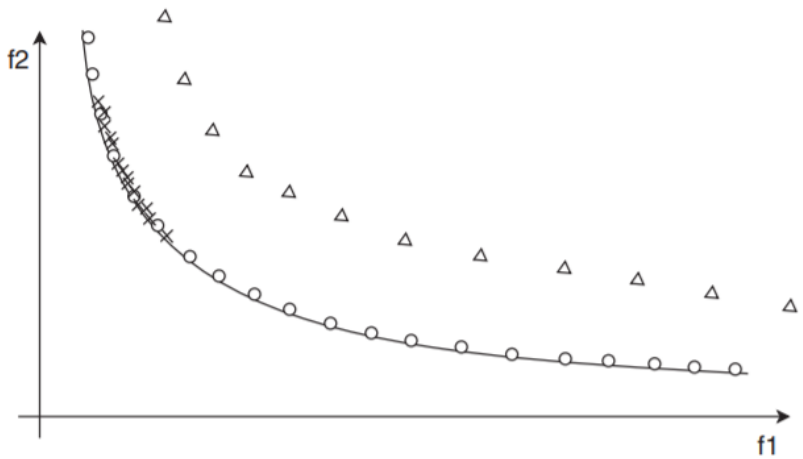


Abbildung 3.3.: Beispiel einer Pareto-Front zweier Zielfunktionen f_1 und f_2

Quelle: [vgl. TBNA12, S. 286]

Die ausgewählten Metaheuristken entsprechen zum Zeitpunkt der Thesis den State-of-the-Art Lösungsverfahren für das MRCPSP. Es werden die im Kapitel 2.2 vorgestellten Metaheuristiken, angelehnt an den bestehenden Publikationen für das (M)RCPSP gemäß der Problemstellung implementiert und verglichen. Des Weiteren werden die Metaheuristiken mit einem Random Solver und dem Hill Climbing-Algorithmus verglichen, welcher der naive lokale Suche entspricht:

Random Solver Der Random Solver entspricht einer naiven Variante und erstellt ohne Heuristiken über das S-SGS zufällige Aktivitäts- und Modilisten. Der Random Solver dient für die Evaluation als eine Referenz, um die Ergebnisse der Metaheuriken besser vergleichen und interpretieren zu können.

Hill Climbing / Local Search (LS) Der generische LS-Algorithmus gilt als eine weitere Referenz zwischen dem Random Solver und den Metaheuristiken. Iterativ wird über die Nachbarschaftsfunktion einer bestehenden Lösung die nächstbeste ausgewählt (vgl. Kapitel 2.2).

Tabu Search (TS) Die TS wurde in unterschiedlichen Versionen bereits 1999 von [KH98, S. 9] für das RCPSP aufgegriffen. Eine Variante verwendet als Nachbarschaft einer Lösung eine Menge von ausgehenden Lösungen mit einzelnen Aktivitätstauschen [vgl. TS98, S. 5].

Simulated Annealing (SA) Abermals wurde 1999 von [KH98, S. 9] für das RCPSP die Simulated Annealing-Metaheuristik aufgegriffen. Hierbei wurde, wie bei der TS, die Nachbarschaft aus einzelnen Aktivitätstauschen realisiert.

Genetic Algorithm (GA) Ebenfalls wurden in der Publikation von [KH98, S. 9] für das RCPSP eine Vielzahl an genetischen Algorithmen verglichen. Bei den Crossover Operatoren steht eine Vielzahl zur Verfügung: Darunter der One-Point Crossover, die Erweiterung, der Two-Point Crossover und die im Abschnitt 2.2.3 bereits vorgestellte Uniform Crossover Operation [vgl. Har98, S. 4 f.].

Als Mutation Operator können austauschbare Aktivitäten innerhalb einer Aktivitätsliste mit einer gewissen Wahrscheinlichkeit $p_{mutation}$ miteinander gewechselt werden. [vgl. Har98, S. 5]

Für das MRCPSP können sowohl der Two-Point Crossover als auch der Mutation Operator genutzt werden [vgl. RSMA15, S. 10 ff.]. Innerhalb dieser Thesis werden beide Operationen zur Realisierung des genetischen Algorithmus verwendet und im Abschnitt 4.3.5 erläutert.

Algorithmen, wie die LS, TS und SA sehen alle eine Nachbarschaftsfunktion $N(s)$ vor. Diese wird innerhalb der Arbeit für alle Verfahren gleichermaßen implementiert.

Aufgrund der möglichen hohen Anzahl an Kombinationen innerhalb einer Nachbarschaft, soll nur eine Teilmenge $\tilde{N}(s) \subset N(s)$ betrachtet werden. Dies geht aus der Publikation von [TS98, S. 5] hervor.

Eine Publikation für das Lösen des MRCPSP mithilfe von Simulated Annealing beschreibt drei Möglichkeiten zur Bildung einer Nachbarschaftsfunktion. Aktivitäten können innerhalb der Aktivitätsliste samt deren Modi verschoben werden. Zudem existiert die Möglichkeit, dass eine Aktivität zufällig selektiert wird und der zugehörige Modus auf einen anderen Modus geändert werden kann. Die dritte Variante stellt die Kombination der beiden vorherigen Varianten dar, welche in dieser Arbeit als Nachbarschaftsfunktion aufgegriffen wird. [vgl. JMR+01, S. 145]

3.4. Benchmarkdatensatz

Die zu implementierenden Metaheuristiken und proaktiven, prädiktiven und reaktiven Verfahren gilt es auf eine gemeinsame Datenbasis für das MRCPSP zu validieren. Unzählige wissenschaftliche Arbeiten, wie die von [RSMA15], [JMR+01] oder [AFH05] nutzen zur Evaluierung ihrer Ergebnisse für das MRCPSP die PSPLIB von [KS97]. Dort enthalten sind Instanzsets, welche neben unterschiedlich generierten Projektplänen auch die Informationen zu den minimalen Projektdauern beinhalten.

	m1	m2	n0	n1	j20
Instanzen	640	481	470	637	554
Aktivitäten	16	16	10	16	20
Modi	1	2	3	3	3
Erneuerbare Ressourcenarten	2	2	2	2	2
Nicht erneuerbare Ressourcenarten	2	2	0	1	2

Tabelle 3.1.: Metadaten zu den verwendeten Instanzsets der PSPLIB [KS97]

Quelle: Eigene Darstellung

Tabelle 3.1 repräsentiert fünf Instanzsets der PSPLIB von [KS97]. Im Rahmen der Evaluierung werden diese verwendet, um die Metaheuristiken miteinander vergleichen zu können. Durch die unterschiedlichen Anzahlen von Aktivitäten, Modi und (nicht) erneuerbaren Ressourcenarten werden die Metaheuristiken mit andersartigen komplexen Gegebenheiten evaluiert. Die PSPLIB beinhaltet zudem das Instanzset j30.mm mit 30 Aktivitäten, in welcher die angegebenen minimalsten Projektdauern zum Zeitpunkt der Masterarbeit nur heuristisch sind, da globale Optima durch die immense Menge an Permutationen nicht bestimmt werden können.

Eine konkrete Instanz n01_2.mm der PSPLIB vom Benchmarkset n0 kann aus Listing 3.1 entnommen werden. Einzelne Sektionen zur Beschreibung des Projektplans werden über Sternchen- und Minuszeichen voneinander getrennt. Die ersten drei Sektionen geben Metainformationen, wie die Anzahl der Aktivitäten, Ressourcenarten, maximale Dauer über Horizon, und Daten zur Generierung der Instanz an. Der vierte Bereich zeigt zum einen die Abhängigkeiten der Aktivitäten und zum anderen die Anzahl der möglichen Modi für eine Aktivität auf. Die Modi werden in dem fünften Bereich mit einer Ausführdauer und den Ressourcenanforderungen definiert. Im sechsten und letzten Bereich der Datei werden die für das Projekt verfügbaren Ressourcen aufgeführt.

Listing 3.1: Instanz n01_2.mm der PSPLIB n0 (Quelle: [KS97])

```
1  ******************************************************************************
2  file  with basedata            : me1_.bas
3  initial  value random generator: 466396357
4  ******************************************************************************
5  projects                       :  1
6  jobs (incl. supersource/sink ):  12
7  horizon                        :  65
8  RESOURCES
9    - renewable                  :  2   R
10   - nonrenewable               :  0   N
11   - doubly constrained         :  0   D
12 ******************************************************************************
13 PROJECT INFORMATION:
14 pronr.  #jobs rel.date duedate tardcost  MPM-Time
15     1     10      0        8        6        8
16 ******************************************************************************
17 PRECEDENCE RELATIONS:
18 jobnr.    #modes  #successors   successors
19    1        1          3           2   3   4
20    2        3          3           5   7   9
21    3        3          2           8   9
22    4        3          3           5   6   7
23    5        3          2          10  11
24    6        3          2           8   9
25    7        3          2          10  11
26    8        3          2          10  11
27    9        3          1          12
28   10        3          1          12
29   11        3          1          12
30   12        1          0
31 ******************************************************************************
32 REQUESTS/DURATIONS:
33 jobnr. mode duration  R 1  R 2
34 ------------------------------------------------------------------------
35   1      1     0       0    0
36   2      1     1       0   10
37          2     6       6    0
38          3     9       0    8
39   3      1     3       4    0
40          2     5       0    7
41          3     7       0    4
42   4      1     1       7    0
```

```
43          2     5       0    10
44          3     8       5     0
45    5     1     1       0     5
46          2     2       0     4
47          3     6       2     0
48    6     1     2       0     3
49          2     3       8     0
50          3     6       7     0
51    7     1     6       6     0
52          2     8       4     0
53          3     9       3     0
54    8     1     1      10     0
55          2     2       8     0
56          3     3       0     6
57    9     1     1       7     0
58          2     4       6     0
59          3     4       0     6
60   10     1     1       6     0
61          2     1       0     7
62          3     3       0     6
63   11     1     1       0     3
64          2     8       4     0
65          3    10       3     0
66   12     1     0       0     0
67 ************************************************************************
68 RESOURCEAVAILABILITIES:
69   R 1  R 2
70     7    7
71 ************************************************************************
```

In Abbildung 3.4 lässt sich der Projektplan für die Instanz n01_2.mm grafisch visualisieren. Diese Darstellungsart ist angelehnt an die aus Abbildung 2.3.

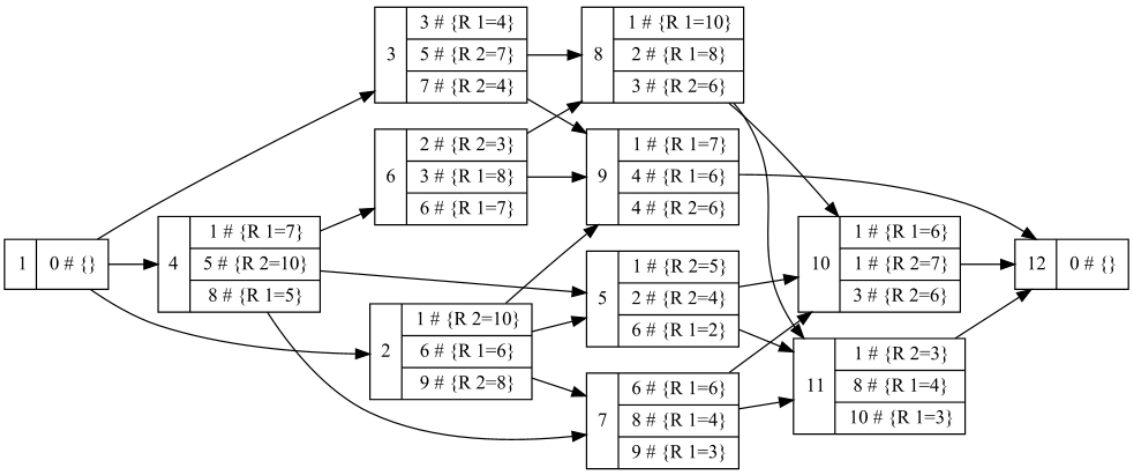


Abbildung 3.4.: Visualisierung des Projektplans der Instanz n01_2.mm der PSPLIB n0

Quelle: Eigene Darstellung

3.5. Unsicherheitsszenarien

Unsicherheiten, wie Aktivitäts- und Ressourcenstörungen müssen zunächst konzipiert und anschließend implementiert werden. Hierfür gilt es die Unsicherheiten in dem existierenden Benchmarkdatensatz aus Abschnitt 3.4 zu integrieren. In dieser Masterthesis werden die Aktivitätsstörungen als einzige Unsicherheitsart analysiert.

Um Unsicherheitsszenarien zu erzeugen, stehen unterschiedliche Arten für die Umsetzung zur Verfügung. [KC13] verwendet für den Vergleich von Robustheitsmessungen ein Störungsschema mit $\Delta_j = 0.20 \cdot d_j$ [vgl. KC13, S. 260]. Folglich verspätet sich jede Aktivität um 20%.

Durch eine fixe Verspätung über $\Delta_j = 0.2 \cdot d_j$ wäre somit jede Aktivität verspätet. Bei dem reaktiven Ansatz, wird zu einem Verspätungszeitpunkt ein neuer Zeitplan generiert. Dies führt dazu, dass nach jeder Aktivität ein neuer Zeitplan generiert werden muss. Bei einer hohen Anzahl an Aktivitäten kann diese Art von Störungsschema zeitaufwendig sein, da ohnehin das Erzeugen von neuen Zeitplänen bei den reaktiven Methoden sich als zeitintensiv herausgestellt hat (vgl. Abschnitt 2.1.4.2). Zudem ist die Annahme, dass jede Aktivität sich um einen konstanten Faktor 20% verspätet nicht zwingend realitätsnah.

Im Bezug auf die Aktivitätsstörungen nutzt [DDH11] die stetige Gleichverteilung, welche eine zufällige Verspätung zwischen $[1, d_j]$ auswählt [vgl. DDH11, S. 72]. Somit entspricht das Unsicherheitsszenario gemäß [DDH11] bei $\Delta_j = \mathcal{U}(1, d_j)$.

Im Rahmen dieser Masterarbeit sollen Verspätungen ebenfalls gemäß einer Verteilungsfunktion erzeugt werden. Dadurch, dass die Dauer $d_{j,m} \in \mathbb{N}$ einer natürlichen Zahl entspricht, sind diskrete Verteilungsfunktionen erforderlich. Die Binomialverteilung $\mathcal{B}(n, p)$ stellt das diskrete Pardon der Normalverteilung $\mathcal{N}(\mu, \sigma)$ dar, welche sich durch die Parametrisierung über die Versuchsanzahl n und dem Wahrscheinlichkeitswert eines Vorkommens p für das Generieren von Unsicherheitsszenarien eignet. Die Neigung einer Binomialfunktion wird über p bestimmt. Bei $p > 0.5$ ist die Verteilung linksschief, bei $p < 0.5$ rechtsschief, ansonsten mit $p = 0.5$ symmetrisch.

Die Parametrisierung der Binomialverteilung $\mathcal{B}(n, p)$ erfolgt über das Berücksichtigen von Definitionen von Annahmen. Die erste Annahme ist, dass eine Aktivität sich um maximal zwei Zeiteinheiten verspäten darf. Folglich wird $n \in \mathbb{N}$ mit zwei definiert, da neben den zwei Verspätungsfällen auch der Fall von keiner Verspätung über das nullte Element in der Verteilung abgedeckt wird. Zudem wird angenommen,

dass ein Großteil der Aktivitäten pünktlich, ein kleinerer Teil mit einer Zeiteinheit und ein noch kleinerer Teil mit zwei Zeiteinheiten Verspätung abgeschlossen werden sollen. Da somit für den Parameter $p \in \mathbb{R}$ unzählige Möglichkeiten zur Verfügung stehen und zudem auch für die Intensität der Unsicherheiten ausschlaggebend ist, werden unterschiedliche Werte für p zur Evaluierung verwendet, welche in Tabelle 3.2 aufgeführt sind.

	$p(\mathcal{B}(2,p)=0)$	$p(\mathcal{B}(2,p)=1)$	$p(\mathcal{B}(2,p)=2)$
p = 0%	100%	0%	0%
p = 5%	90.25%	9.5%	0.25%
p = 10%	81%	18%	1%
p = 20%	64%	32%	4%

Tabelle 3.2.: Wahrscheinlichkeitswerte für Aktivitätsstörungen gemäß der Binomialfunktion $\mathcal{B}(3,p)$

Quelle: Eigene Darstellung

Durch das Erzeugen von Unsicherheitsszenarien für Aktivitätsstörungen über die Binomialfunktion $\Delta_j = \mathcal{B}(n,p)$ ist der Zufall ausschlaggebend. Insbesondere bei der Evaluierung der pro-/prä-/reaktiven Verfahren sollten mehrere Unsicherheitsszenarien für einen zu betrachtenden Zeitplan S durchlaufen werden. Erst durch das Durchlaufen unterschiedlicher Unsicherheitsszenarien können Aussagen über die tatsächlichen Leistungen der Verfahren gemacht werden. Dieses Problem ist bei dem deterministischen Ansatz über $\Delta_j = 0.2 \cdot d_j$ nicht vorzufinden, da im Vornherein alle Aktivitäten die gleiche Verspätung aufweisen. Zunächst ist die Berechnung mehrerer unterschiedlicher Experimente pro Zeitplan S fraglos zeitaufwendiger, können aber heutzutage in der CPU bei entsprechender Anzahl der Kerne parallel ausgeführt werden.

Kapitel 4

Implementierung des MRCPSP-Frameworks

In diesem Kapitel wird die Implementierung des MRCPSP-Frameworks einschließlich der entwickelten Metaheuristiken vorgestellt. Zunächst gilt es die Struktur des Frameworks im Abschnitt 4.1 basierend auf einem UML-Klassendiagramm aufzuzeigen. Das Thema der Zeitplanerstellung, Heuristiken und die Visualisierung wird im Folgeabschnitt behandelt. Abschnitt 4.3 stellt die unterschiedlich implementierten Lösungsansätze und Metaheuristiken vor. Der Aufbau von Experimenten, wie beispielsweise der Vergleich von den Lösungsansätzen oder die Handhabung von Unsicherheitsszenarien gilt es im Abschnitt 4.4 zu vertiefen.

4.1. Strukturbeschreibung

Wie bereits im Abschnitt 3.1 erläutert wird für die Entwicklung des MRCPSP Framework die objektorientierte Programmiersprache Java einschließlich des Spring genutzt. Über die objektorientierte Programmierung können Teilaspekte der Anwendung über Klassen voneinander getrennt werden. Packages können zudem eingesetzt, um Aspekte voneinander zu trennen, um so den Code übersichtlicher zu halten. Innerhalb des MRCPSP Frameworks sind die Packages nach Kategorien angelegt. Folgende Paketstruktur ist mit entsprechenden Funktionen in der Umsetzung vorgesehen:

Rootpackage de.uol.sao.rcpsp_framework beinhaltet die Mainklasse und somit den Einstiegspunkt der Anwendung. Durch die Verwendung vom Spring Framework werden weitere Einstiegspunkte in der Anwendung über die Komponenten/Services Annotation festgelegt.

exception Eigene Exceptions die innerhalb der Anwendung auftreten sind in diesem Package enthalten.

experiment Einzelne zu untersuchende Aspekte werden über Experimente geregelt.

heuristic Klassen zur Generierung von Aktivitäts- und Moduslisten anhand bestimmter Prioritäts- und Modusregeln.

metric Unterschiedliche Funktionen für Zeitplan-Metriken, um bestimmte Informationen, wie die Projektdauer oder Robustheit zu berechnen.

function Zielfunktionen, die entweder für einen direkten Vergleich zweier Zeitpläne oder für die Fitness-Berechnung eines einzelnen Plans eingesetzt werden.

representation Unterschiedliche Representationsformen von Aktivitäts- und Modusselektionen für Zeitpläne die innerhalb des MRCPSP Frameworks verwendet werden können.

scheduling Klassen zum Erstellen von Zeitplänen und deren Informationen.

service Mit `@Service` annotierte Klassen, welche in der kompletten Anwendung an unterschiedlichen Stellen verwendet werden.

solver Beliebige Lösungsverfahren u. a. Metaheuristiken, die in der Anwendung verwendet werden.

helper Hilfsklassen und deren Methoden, die über die komplette Anwendung genutzt werden und sich nicht in anderen Packages klassifizieren lassen.

Durch die Implementierung entstand das zusammengefasste UML-Klassendiagramm aus Abbildung 4.1, welches aufgrund der Komplexität keine Generalisierungen von abstrakten Klassen und Interfaces berücksichtigt. Im Verlauf des Kapitels werden einzelne UML-Klassendiagramme zur Erläuterung der Umsetzung eingesetzt und erläutert, welche diese Generalisierungen aufführt.

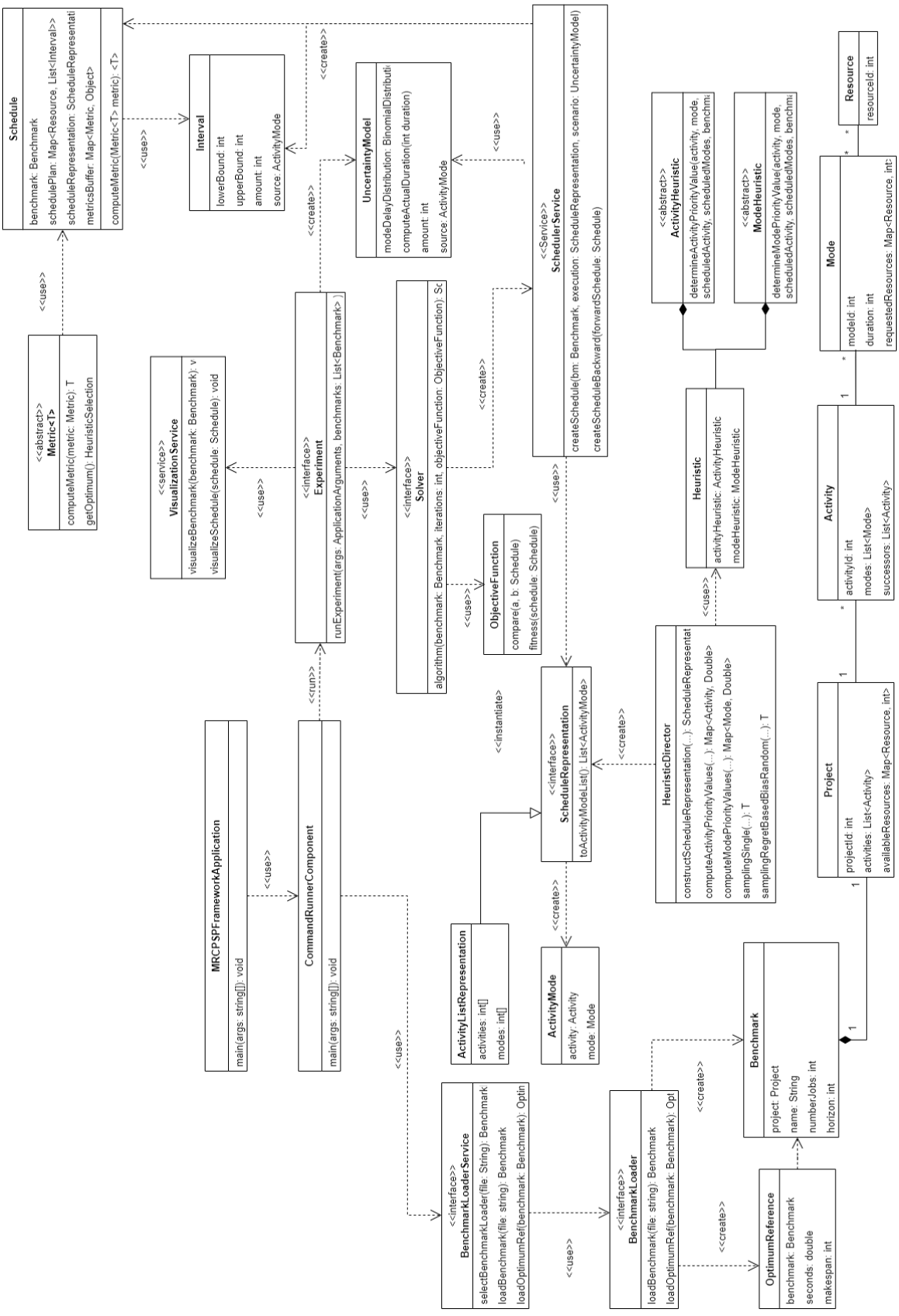


Abbildung 4.1.: Zusammengefasstes UML-Klassendiagramm des MRCPSP-Framework

Quelle: Eigene Darstellung

4.2. Zeitplanerstellung und Heuristiken

Um Zeitpläne für das MRCPSP zu erstellen, gilt es die zugehörigen Constraints aus Abschnitt 2.1 einzuhalten. Hierfür ist im Konzeptüberblick die Komponente Scheduler vorgesehen. Des Weiteren müssen Lösungsansätze und Verfahren zur Lösung von Unsicherheiten auf eine gemeinsame Datenbasis verglichen werden, um für diese Zeitpläne erstellen zu können. Hierfür sind Benchmark Loader und Benchmark Instance Sets aus dem Abschnitt 3.1 vorgesehen.

4.2.1. Benchmarkinstanzen

Zum Laden der Projektinstanzen von der PSPLIB ist ein BenchmarkLoaderService implementiert, welcher anhand der Datenendung einen passenden Loader selektiert. Im Falle der PSPLIB für die Multi-Mode Projekterweiterung wird die Datenendung .mm verwendet [vgl. KS97, S. 213 f.]. Der vorgesehene BenchmarkPSPLIBMultiModeLoader lädt aus der PSPLIB die vorgesehenen Struktur das Projekt einschließlich Metadaten, Aktivitäten, Aktivitätsbeziehungen, Modi und die Ressourcenanforderungen aus und überträgt diese innerhalb der im MRCPSP Framework vorgesehenen Plain Old Java Objects (POJOs). Die einzelnen POJOs orientieren sich hierbei an den Subjekten und deren Beziehungen von der MRCPSP-Problembeschreibung aus Abschnitt 2.1.2. Abbildung B.1 zeigt die Klassen des Benchmark Loaders und die der POJOs in einem UML-Klassendiagramm auf.

4.2.2. Zeitplanerstellung

Für die Erstellungen von MRCPSP-Zeitplänen können unterschiedliche Repräsentationsformen verwendet werden. Neben den Aktivitäts- und Moduslisten existiert die Random Key Representation, welche in den Publikationen von [SAA15] und [KH98] verwendet wird. Diese Arbeit beschränkt sich jedoch nur auf die Aktivitäts- und Moduslisten, welche in den Abschnitten 2.1.3.2 und 2.1.3.3 eingeführt wurden. Hierfür ist eine Klasse ActivityListRepresentation vorgesehen, die von der Klasse ScheduleRepresentation erbt und zwei Felder für die Aktivitäten und für die Modi vorsieht. Diese Repräsentationsform wird von den (Meta-)Heuristiken zur Erstellung von Schedules genutzt. Abbildung B.2 stellt das UML-Klassendiagramm für die Realisierung der Repräsentationsform, aber auch der Zeitplanerstellung und Heuristiken dar.

Der SchedulerService erzeugt mit der Methode createSchedule(...) Zeitpläne anhand von Aktivitäten- und Modirepräsentationen $I = (\lambda, \mu)$ gemäß des S-SGS. Hierbei werden die Ressourcenbelegungen über Intervalle (Interval) festgehalten. Listing

C.1 zeigt den Algorithmus für das sequenzielle Abarbeiten der Aktivitäten und den zugehörigen Modi (ActivityMode) auf. Für jeden ActivityMode wird ein identischer Intervall auf alle zu verwendenden erneuerbaren Ressourcen erzeugt. Intervalle geben an, wie lange eine erneuerbare Ressource mit wie vielen Einheiten belegt ist. Sei I ein Intervall mit $I = [a, b]$, dann ist a die Startzeit und b die Endzeit einer Aktivität mit Bezug auf die ausgewählte Modusdauer. Im Algorithmus wird a mit potentialLowerBound und b mit potentialUpperBound gesetzt. Es gilt, dass eine Aktivität erst gestartet werden kann, wenn alle Vorgänger komplett durchlaufen wurden (vgl. Abschnitt 2.1.1). Zudem müssen Aktivitäten ohne Unterbrechungen mit der Menge an benötigten Ressourcen ausgeführt werden (vgl. Abschnitt 2.1.1).

Im Algorithmus wird zunächst die initiale Startzeit potentialLowerBound gemäß der Vorgängerbeziehungen selektiert. Es gilt zu überprüfen, ob für die Aktivität zum Zeitpunkt ausreichend Ressourcen vorliegen. Hierfür wird eine Schleife durchlaufen, die für jede Ressourcenart gemäß der (Modus-)Ressourcenanforderungen überprüft werden, ob diese in benötigter Zahl $r_{j,m,k} \wedge c_{j,m,l}$ vorliegen (vgl. Listing C.2). Wenn dies der Fall ist, wird die Schleife mittels solutionFound = **true** verlassen und das gefundene Intervall für alle relevanten Ressourcenarten im Zeitplan hinzugefügt. Wenn bei einer erneuerbaren Ressourcenart die benötigte Anzahl nicht vorliegt, wird ein neues Interval erzeugt, indem potentialLowerBound und potentialUpperBound um eins inkrementiert werden. Anschließend wird das neue Intervall erneut für alle benötigten Ressourcenarten überprüft. Diese Prozedur wird solange ausgeführt, bis ein geeignetes Intervall gefunden wurde. Sofern die Anzahl an benötigten erneuerbaren Ressourcen die generelle Verfügbarkeit überschreitet oder wenn bereits alle nicht-erneuerbaren Ressourcen verbraucht wurden, handelt es sich um einen ungültigen Plan und somit wird der Algorithmus über das Exception-Handling abgebrochen. Bei Durchlaufen aller Einträge der Aktivitäts- und Moduslisten entsteht somit für die Repräsentation $I = (\lambda, \mu)$ ein minimaler Zeitplan.

Über einen Zeitplan können jegliche Metriken berechnet werden. Hierfür ist eine abstrakte Klasse Metric<?> vorgesehen. Diese wird von den konkreten Metriken geerbt. Die relevanteste Metrik stellt die Makespan dar, welche sich über die höchste Endzeit aller ermittelten Intervalle ableiten lässt. Weitere Metriken können die Robustheitsmessungen darstellen. Diese wiederum nutzen meist den freien Puffer von Aktivitäten, welche sich über das Backward Recursive Procedure berechnen lassen (vgl. Abschnitt 2.1.4.1).

Während beim gewöhnlichen Forward Recursive Procedure die frühstmöglichen Start- und Endzeitpunkte gemäß der Aktivitäts- und Modusliste bestimmt werden,

wird beim Backward Recursive Procedure eine Aktivität zum spätestmöglichen Zeitpunkt geschedulded [vgl AFH05, S. 181]. Somit ist es möglich, die spätesten Start- und Endzeitpunkte von Aktivitäten zu bestimmen, um so den Puffer einer Aktivität berechnen zu können [vgl AFH05, S. 181]. Dieser Algorithmus wird ebenfalls im SchedulerService über die Methode createScheduleBackward() realisiert.

4.2.3. Heuristiken

Eine Möglichkeit Aktivitäts- und Moduslisten $I = (\lambda, \mu)$ zu erzeugen, stellen zunächst die Heuristiken dar. Diese werden innerhalb der Arbeit verwendet, um initiale Zeitpläne für die Metaheuristiken zu erzeugen. Als Heuristiken werden die prioritätsbasierten Regeln für Aktivitäten (gemäß Tabelle 2.2) und die Selektionsregeln für Modi (gemäß Tabelle 2.4) über den S-SGS eingesetzt.

Die zu implementierenden Prioritätsregeln erben von der Klasse ActivityHeuristic, die eine abstrakte Methode determineActivityPriorityValue(...) besitzt. Die konkretisierende Klasse implementiert die Methode anhand der Prioritätsregel und gibt den Prioritätswert für eine Aktivität Activity zurück. Innerhalb der Abbildung B.2 lassen sich somit die konkreten Klassen aus Tabelle 2.2 einschließlich einer Zufallsregel erkennen:

- GRPWHeuristic
- LFTHeuristic
- LSTHeuristic
- MSLKHeuristic
- MTSHeuristic
- RandomActivityHeuristic

Das Prinzip gilt ebenfalls für die zu implementierenden Selektionsregeln, welche die Klasse ModeHeuristic vorsieht. Über die Methode determineModePriorityValue(...) wird der Selektionswert für eine Aktivität mit einem bestimmten Modus berechnet. Abbildung B.2 zeigt die konkreten Klassen für die implementierten Selektionsregeln einschließlich einer Zufallsregel auf:

- LPSRDHeuristic
- LRSHeuristic
- LTRUHeuristic

- SMFHeuristic

- RandomModeHeuristic

Heuristic sieht jeweils eine konkrete ActivityHeuristic und ModeHeuristic vor und ermöglicht somit etwaige Kombinationen zwischen den Regeln. Dreh und Angelpunkt für die Erzeugung von heuristischen Lösungen stellt der HeuristicDirector. Die Methode constructScheduleRepresentation(...) erzeugt anhand eines Objektes der Klasse Heuristic und einer Angabe des Sampling-Verfahrens entsprechende Instanzen vom Typ ScheduleRepresentation, welche konkret die Aktivitäts- und Moduslistenrepräsentation darstellen. Dies geschieht, indem Aktivitäten und Modi gemäß des S-SGS von Abschnitt 2.1.3 sequenziell selektiert werden. Beim Sampling stehen die folgenden Möglichkeiten zur Verfügung:

Single Aktivitäten und Modi werden gemäß ihres Prioritäts- bzw. des Selektionswertes selektiert. Zunächst wird in einer Phase g für alle möglichen Aktivitäten $\mathcal{D}_g$ jeweils ein Modus gemäß des besten Selektionswertes $s(j,m)\,\forall j \in \mathcal{D}_g$ ausgewählt. Im Anschluss wird die Aktivität einschließlich des selektierten Modus über den Prioritätswert $v(j)$ selektiert.

Regret Based Biased Random Sampling (RBRS) Aktivitäten und Modi gilt es ebenfalls über den Prioritäts- bzw. Selektionswert auszuwählen, jedoch werden hierfür Wahrscheinlichkeiten vorgesehen, um so abweichend mehrere Lösungen zu erhalten. Das Prinzip von RBRS als Multi-Pass Sampling Methode wurde bereits im Abschnitt 2.1.3.4 erläutert. Der Parameter ϵ wird dem Wert $\epsilon = \max(0.001, \min_{i \in \mathcal{D}_g} v(i))$ zugewiesen, welche die Empfehlung von Drexl (1991) gemäß [KH98, S. 7] dahingehend modifiziert, sodass Epsilon nie null ist. Dies verhindert invalide Divisionen. Für den Parameter α wird empirisch der konstante Wert $\alpha = 10$ festgelegt. Insbesondere bei Benchmarks mit hohem nicht-erneuerbarem Ressourcenverbrauch und einer hohen Anzahl an Aktivitäten können mithilfe der Selektionsregel LRSHeuristic mit hoher Wahrscheinlichkeit zunächst gültige Zeitpläne erstellt werden, da die noch benötigten nicht-erneuerbaren Ressourcen in Relation zu den verfügbaren nicht-erneuerbaren Ressourcen gestellt werden. Es wird der Modus ausgewählt, für welcher in Relation die meisten Ressourcen vorhanden sind. Insbesondere diese Selektionsregel profitiert von einem hohen α-Parameter, um so eher stärker an der Regel orientierte Lösungen zu erhalten, aber trotzdem gewisse Abweichungen zu erlauben.

4.2.4. Visualisierung

Innerhalb der Anwendung werden Konsolenausgaben genutzt, um so den Fortschritt und die Ergebnisse der Experimente auszugeben. Nach Durchlaufen eines Lösungsansatzes wird der beste Zeitplan einschließlich der Aktivitäts- und Modusliste, Makespan und ggf. der Robustheitsmetrik ausgegeben. Dadurch können Zeitpläne reproduziert werden. Hierfür wird beim Initialisieren des MRCPSP-Frameworks ein Thread erzeugt, welcher parallel zu den Berechnungen läuft. Dieser wird über den Service CommandLineToolService implementiert und nimmt den Benchmarknamen, die Aktivitätsliste und die Modusliste einzeln entgegen und erstellt über den Scheduler einen manuellen Zeitplan. Dieser wird sowohl in Textform als auch grafisch visualisiert.

Viele Teilbereiche dieser Arbeit werden über Zahlen verdeutlicht. Um jedoch ein Bild von erstellten Zeitplänen machen zu können eignet sich die Möglichkeit der grafischen Visualisierung. Hierfür ist der VisualizationService vorgesehen, welcher den Projektplan eines Benchmarks, aber auch zugehörige Zeitpläne über die Ressourcenbelegungen grafisch darstellen kann.

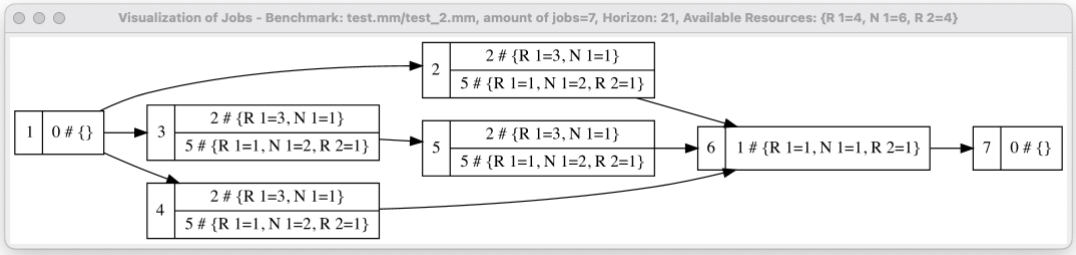


Abbildung 4.2.: Screenshot einer Projektplan-Visualisierung

Quelle: Eigene Darstellung

Abbildung 4.2 stellt einen Screenshot von der Visualisierung eines beispielhaften MRCPSP-Projektplans über den VisualizationService dar. Die Darstellungsweise wurde bereits im Abschnitt 2.1.2 erläutert. Für die Generierung von Projektplänen wird die Software Graphviz von [AB21] verwendet. visualizeBenchmark(benchmark) visualisiert einen Projektplan, indem jede Aktivität einen Knoten repräsentiert und diese gemäß ihrer Nachfolgebeziehungen verbunden werden. Die Positionierung der Knoten erfolgt über Graphviz automatisch.

(a) (b)

Abbildung 4.3.: Screenshots einer Zeitplan-Visualisierung über erneuerbare Ressourcenarten für den Projektplan aus Abbildung 4.2

Quelle: Eigene Darstellung

Abbildung 4.3 zeigt Screenshots für die Darstellung von Zeitplänen auf. Hierbei stellen Abb. 4.3a und Abb. 4.3b den Ressourcenverbrauch für die erneuerbare Ressourcenart R_1 und R_2 dar. Die vertikalen Balken stellen den Ressourcenverbrauch von Aktivitäten dar, welche über die Farbe differenziert werden. Das maximale Ressourcenlimit für eine Ressource R_k wird über die horizontale schwarze Linie angegeben. Folglich darf diese gemäß der MRCPSP-Beschränkung aus Formel 2.7 nicht überschritten werden.

Für den nicht-erneuerbaren Ressourcenverbrauch wird die gleiche Darstellung wie bei dem erneuerbaren Ressourcenverbrauch verwendet. Jedoch werden die nicht-erneuerbaren Ressourcen bis zum Ende des Zeitplans verwendet. Dies wird in Abbildung 4.4 verdeutlicht.

Für die Visualisierung von Zeitplänen wird die Methode visualizeSchedule(schedule) aus der Klasse VisualizationService verwendet. Hierbei werden die XYPlot aus der Bibliothek JFreeChart von [Gil21] verwendet, um die vertikalen Balken aufeinander stapeln zu können.

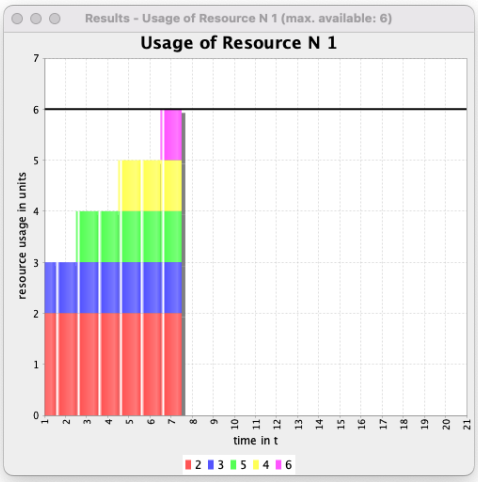


Abbildung 4.4.: Screenshot einer Zeitplan-Visualisierung über nicht-erneuerbare Ressourcenarten für den Projektplan aus Abbildung 4.2

Quelle: Eigene Darstellung

4.3. Lösungsansätze

Bereits im Abschnitt 3.3 wurden Metaheuristiken und andere Lösungsansätze selektiert, um Zeitpläne für das MRCPSP zu finden. Als abstrakte Anlaufstelle für die Lösungsverfahren dient das Interface Solver, welches eine Methode algorithm(...) vorsieht. Dieses wird parametrisiert über ein Benchmark-Objekt, die Anzahl an Iterationen, eine Robustheitsfunktion und eine feste Anfangsliste von Aktivitäten- und Modi, welche wiederum relevant für die reaktiven Verfahren sind. Aufgerufen wird die Methode über die einzelnen Experimente. Die konkreten Lösungsansätze implementieren diese Methode gemäß des eigentlichen Algorithmus. Für den fairen Vergleich der Algorithmen werden die Iterationen als ein Zähler von erzeugten Aktivitäts- und Moduslisten betrachtet. Das Erzeugen eines Tupels erhöht die durchlaufene Iterationen um eins. Bei einer Nachbarschaft aus fünf Elementen dementsprechend um fünf.

Das UML-Klassendiagramm aus Abbildung B.3 zeigt das Zusammenspiel der Lösungsverfahren genauer auf. Alle aus Abschnitt 3.3 selektierten Lösungsansätze werden über eine eigene Klasse repräsentiert, welche das Interface Solver implementieren. Diese erzeugen über die jeweiligen Algorithmen die Aktivitäts- und Moduslisten, um so Zeitpläne über den SchedulerService zu erzeugen. Insbesondere für die initialen Lösungen nutzen die Metaheuristiken den HeuristicDirector, um so machbare oder sogar gute Lösungen zunächst heuristisch zu generieren. Insbesondere der GeneticAlgorithm nutzt für die initiale Population unterschiedliche Prioritäts- und Selektionsregeln.

Ebenfalls im Abschnitt 3.3 wurde eine gemeinsame Nachbarschaftsfunktion für sowohl Hill Climbing, Tabu Search als auch Simulated Annealing festgelegt. Die Funktionsweise der implementierten Nachbarschaftsfunktion $\tilde{N}(s)$ ist in Abbildung 4.5 illustriert. Für einen zu betrachtenden Zeitplan s werden stets zwei Zeitpläne in der Nachbarschaft hinzugefügt, welche eine identische Aktivitätsliste zum Basisplan aufweisen. Dennoch wird abweichend zum Basisplan ein Modus innerhalb der Modusliste umgedreht. Alle vom Basisplan aus gültigen Aktivitätsswaps werden ebenfalls einschließlich eines Moduslistenelement-Flip in der Nachbarschaft hinzugefügt.

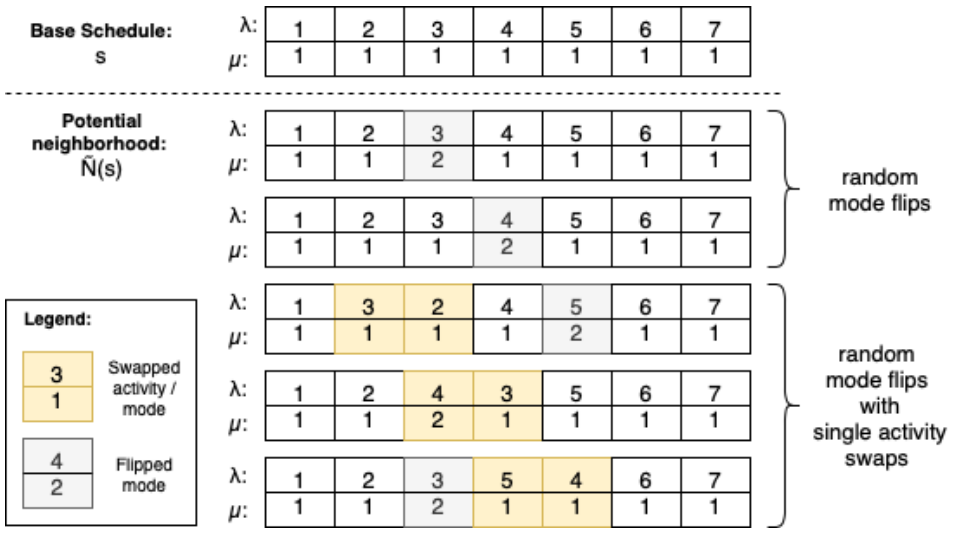


Abbildung 4.5.: Funktionsweise der implementierten Nachbarschaftsfunktion anhand eines Beispiels für den Projektplan aus Abbildung 4.2

Quelle: Eigene Darstellung

In den folgenden Unterabschnitten werden die implementierten Lösungsansätze für das MRCPSP erläutert. Zunächst wird im Abschnitt 4.3.1 die Umsetzung der Erstellung von zufälligen Lösungen vorgestellt. Im Folgeabschnitt 4.3.2 wird der Hill Climbing-Algorithmus behandelt. Die Tabu Suche wird in Ausschnitt 4.3.3 erläutert, den Simulated Annealing-Algorithmus in Abschnitt 4.3.4. Zuletzt gilt es den umgesetzten genetischen Algorithmus im Abschnitt 4.3.5 vorzustellen.

4.3.1. Zufällige Lösungen

Den einfachsten Lösungsansatz stellt das Generieren von zufälligen Lösungen dar. Der RandomSolver generiert mit Hilfe des S-SGS und dem HeuristisDirector zunächst syntaktisch korrekte Zeitpläne. Hierbei werden als Prioritäts- und Selektionsregeln die Klassen RandomActvitiyHeuristic und RandomModeHeuristic eingesetzt, welche zufällige

Zahlen aus dem Intervall $[0, 10000]$ generieren. Innerhalb des Algorithmus wird in jeder Iteration überprüft, ob der gefundene Zeitplan primär die Makespan C_{max} und ggf. sekundär die Robustheitsfunktion Ω verbessert und legt den besten Zeitplan fest. Dieser Algorithmus wird über eine endliche Zahl an Iterationen durchlaufen.

4.3.2. Hill Climbing

Hill Climbing stellt mit der Klasse HillClimbingSolver die Realisierung des naiven LS-Algorithmus aus Abschnitt 2.2 dar.

Der LS-Algorithmus sieht eine initiale Lösung vor, welche heuristisch über zufällige Kombinationen von Prioritäts- und Selektionsregeln erstellt werden kann. Mit einer Wahrscheinlichkeit von 66% wird beim HeuristicDirector die Methode *Single Sampling* ausgewählt, andernfalls werden Aktivitäts- und Moduslisten über *Regret Based Biased Random Sampling* erzeugt. Es werden solange verschiedene Konstellationen von Prioritäts, Selektions- und Samplingverfahren ausprobiert, bis ein gültiger, initialer Zeitplan gefunden wurde. Insbesondere die Selektionsregel LRSHeuristic eignet sich für Projekte mit einer hohen Komplexität seitens der nicht-erneuerbaren Ressourcen. Diese Art der Erstellung von initialen Lösungen wird auch bei der TS und SA angewandt.

Gemäß des erläuterten LS-Algorithmus aus Abschnitt 2.2 und der definierten Nachbarschafts(teil)funktion aus Abschnitt 4.3 wird somit iterativ die beste Lösung ausgewählt bis das lokale Optimum erreicht wurde.

4.3.3. Tabu Search

Bereits im Abschnitt 2.2.1 wurde die Tabu Suche als eine Erweiterung des naiven LS-Algorithmus (bzw. Hill Climbing) eingeführt. Wesentliche Komponenten, wie die generelle Funktionsweise der schrittweisen Verbesserungen, die Nachbarschaftsfunktion oder das Erzeugen der initialen Zeitpläne bleiben gleich. Folglich wurden die Implementierungen der Komponenten für die TS aus Abschnitt 4.3.2 übernommen. Die Implementierung der Tabu Suche wurde in der Klasse TabuSearchSolver realisiert.

Eins der am meist verbreitetsten (Basis-)Konzepte für die Tabu Search stellt die Tabu List dar [vgl. GP19, S. 42]. Dieses wurde im Rahmen der eigenen Implementation aufgegriffen. Die Größe der Tabu List stellt einen Hyperparameter dar, welcher mit $|TL| = \sqrt{|J| - 2}$ versehen wurde. Des Weiteren werden neue Elemente am Anfang der Liste hinzugefügt. Die restlichen Elemente sind anschließend jeweils um eine Position verschoben, wobei das letzte Element von der Liste entfernt wird, sofern

dieses nicht mehr in die Liste passt. Elemente aus der Tabu Liste werden gemäß des Konzepts nicht in der Nachbarschaft in Betracht gezogen [vgl. GP19, S. 42].

4.3.4. Simulated Annealing

Eine Umsetzung des an der Metallurgie angelehnten SA-Algorithmus gilt es in diesem Abschnitt zu erläutern. Die Implementierung des vorgestellten Algorithmus geschieht über die Klasse SimulatedAnnealingSolver. Die folgenden Hyperparameter wurden mit den hinterlegten Werten umgesetzt, welche sich über kleinere Tests als geeignet erwiesen haben:

$T_0 = 1000$

Initiale Temperatur T_0: Für die initiale Temperatur T_0 wurde ein Wert von $T_0 = 1000$ ausgewählt.

Abkühlungsrate α: Die Temperatur wird über die Iterationen mit einer Rate von $\alpha = 0.9$ abgekühlt.

Der implementierte Algorithmus orientiert sich an dem Pseudocode aus Listing 2.4. Neben der Auswahl der Hyperparameter gilt es noch die Zielfunktion $f(x)$ zu konkretisieren und die Realisierung der Auswahl der Nachbarschaftsfunktion anzupassen.

Abweichung zur Nachbarschaftsselektion

Im Pseudocode aus Abschnitt 2.2.2 wird in einer Iteration eine zufällige Lösung aus der Nachbarschaft selektiert. In der Umsetzung für das MRCPSP wird jedoch jedes Element einer Nachbarschaft mit einer Wahrscheinlichkeit von 50% nicht betrachtet. Bei den restlichen zu betrachtenden Elementen einer Nachbarschaft wird das beste Element gemäß einer Funktion $f(x)$ selektiert. Dadurch wird ein Zufall gewährleistet, welcher aber nicht willkürlich die schlechtesten Lösungen selektiert.

Realisierung der Zielfunktion $f(x)$

Für den SA-Algorithmus ist eine Zielfunktion $f(x)$ vorgesehen. Die Fitnessfunktion $f(x)$ des umgesetzten GA-Algorithmus aus Abschnitt 4.3.5 wird ebenfalls für den Algorithmus eingesetzt.

4.3.5. Evolutionäre Algorithmen

Einen komplexen Lösungsansatz stellen die genetischen Algorithmen, ein Teilbereich der evolutionären Algorithmen dar. In diesem Abschnitt wird die Implementierung

eines genetischen Algorithmus über die Klasse GeneticAlgorithmSolver erläutert.

Ein genetischer Algorithmus kennzeichnet sich durch eine Vielzahl an Hyperparameter. Für den implementierten Algorithmus wurden diese empirisch mit Werten definiert, welche sich innerhalb von kleineren Tests als geeignet dargestellt haben:

- **Eltern μ:** Für die Anzahl der Eltern und somit auch die Populationsgröße wurde $\mu = 40$ ausgewählt
- **Nachkommen λ:** Für die Anzahl der Nachkommen und somit auch die der erzeugten Zeitpläne in einer Generation wird mit $\lambda = 50$ selektiert.
- **Lebensdauer κ:** Die maximale Lebensdauer eines Individuums liegt bei $\kappa = 50$ Generationen.
- **Mutationsrate σ:** Die Mutationsrate wird zum Start des Algorithmus bei $\sigma = 0.06$ gesetzt und wird über eine Rechenberg Regel im Verlauf des Algorithmus angepasst. Die Rechenberg Regel wird im Verlauf der Vorstellung des Mutation Operator erläutert.

Bereits im Abschnitt 2.2.3 wurden die unterschiedlichen Phasen über einen generischen genetischen Algorithmus aus Listing 2.5 vorgestellt. Diese gilt es nun für das konkrete Problem des MRCPSP und für den eigenen Algorithmus zum Finden von guten Lösungen umzusetzen.

Initiale Population

Eine initiale Population P mit $\mu = 40$ Individuen wird heuristisch erzeugt, indem verschiedene Aktivitäts- und Selektionsregeln miteinander kombiniert und solange durchlaufen werden, bis die Population vollständig ist. Dies dient dazu, unterschiedliche Lösungen mit den Vorzügen der Heuristiken zu erzeugen. Mit einer Wahrscheinlichkeit von 66% wird als Sampling-Verfahren die Single Pass-Methode eingesetzt, andernfalls die RBRS-Methode. Insbesondere Benchmarks mit einer hohen Komplexität an nicht-erneuerbaren Ressourcen führen dazu, dass das Erzeugen einer Population mehr Iterationen benötigt, wodurch ein Schwellwert eingeführt wird, welcher bei Überschreitung den Selektionsmodus LRSHeuristic erzwingt.

Umsetzung der Auswahl der Eltern

Für die Crossover Operation müssen zunächst zwei Eltern $\rho = 2$ ausgewählt werden. Hierfür werden ρ zufällige Einträge aus der Population entnommen.

Umsetzung des Crossover Operator

Bei dem eingesetzten Crossover Operator handelt es sich um den *Two-Point Crossover*. Hierbei wird ein Tochterelement $D = (\lambda^D, \mu^D)$ erzeugt, wobei λ^D und μ^D die Aktivitäts- und Moduslisten von D darstellen.

[Har98] bezieht sich in der Quelle zur Definition des Two-Point Crossovers auf das Basisproblem und somit nur auf die Aktivitätslistendarstellung. Die Anwendung auf Moduslisten durch Berücksichtigung der Positionierung ist dennoch möglich. Bei dem Two-Point Crossover werden zwei zufällige Punkte q_1 und q_2 gemäß $1 \leq q_1 < q_2 \leq J$ ausgewählt. Zudem sei $\rho_1 = (\lambda^{\rho_1}, \mu^{\rho_1})$ das Mutter- und $\rho_2 = (\lambda^{\rho_2}, \mu^{\rho_2})$ das Vaterelement. i soll nun Positionen der Aktivitäts- und Modusliste aufzeigen. Die Aktivitäten und Modi von dem Mutterelement ρ_1 sollen zwischen $i = 0 \dots q_1$ auch für das Tochterelement D gelten. Als nächster Schritt wird das Vaterelement ρ_2 berücksichtigt. Hierbei werden die Positionen zwischen $i = q1 + 1 \dots q_2$ vom Vater hergeleitet. Das Problem besteht, dass Aktivitäten doppelt auftreten können. Folglich werden Elemente zu D hinzugefügt, für welche ein k existiert, sodass gilt: $\lambda_k^{\rho_2} \notin \{\lambda_1^D, ..., \lambda_{i-1}^D\}$. Zwischen $i = q_2 + 1 \dots J$ wird das gleiche Prinzip wieder für das Mutterelement ρ_1 angewandt. Es werden die restlichen Elemente zu D vom Mutterelement hinzugefügt, für welches ein k existiert, sodass gilt: $\lambda_k^{\rho_1} \notin \{\lambda_1^D, ..., \lambda_{i-1}^D\}$. Die Anwendung des Two-Point Crossover anhand eines Beispiels lässt sich in Abbildung 4.6 illustrieren. [vgl. Har98, S. 5]

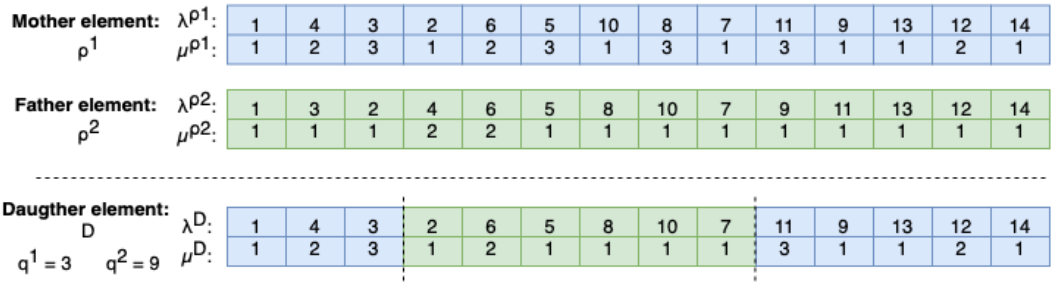


Abbildung 4.6.: Beispiel des Two-Point Crossover für das MRCPSP

Quelle: Eigene Darstellung

Umsetzung des Mutation Operator

Der implementierte Mutation Operator soll innerhalb der Aktivitäts- und Moduslisten von Nachfahren λ leichte Änderungen realisieren. Tupels von austauschbaren Aktivitäten werden hierbei jeweils mit einer Wahrscheinlichkeit von $p_{mutation} = \sigma$ miteinander vertauscht. Mit der selben Wahrscheinlichkeit σ zudem wird für jede Aktivität erneut ein zufälliger Modus selektiert.

Die Wahl einer optimalen Mutationsrate σ stellt eine Herausforderung dar. Die Rechenberg Regel modifiziert die Mutationsrate σ in Abhängigkeit der Erfolgsrate einer Population. Bei einer Erfolgsrate von weniger als 1/5 wird die Rate erhöht, bei 1/5 bleibt die Mutationsrate gleich. Sofern die Erfolgsrate größer als 1/5 ist, wird die Mutationsrate erhöht. [vgl. Kra17, S. 24 f.]

Eine abgewandelte Form der Rechenberg Regel wird für den umgesetzten Algorithmus verwendet. Für die Implementierung bezieht sich die Erfolgsrate auf die Verbesserung des besten Individuums. Eine Population, die das neue beste Individuum gemäß der Fitness beinhaltet, wird als erfolgreich angesehen. Die Mutationsrate σ wird über die folgende Formel geändert:

$$\sigma = \sigma * \exp^{0.05}(a - \tfrac{1}{5}) \qquad \text{mit } a = \begin{cases} 0 & \text{mit Erfolgsrate} < 1/5 \\ 1 & \text{mit Erfolgsrate} \geq 1/5 \end{cases}$$

Umsetzung des Selection Operator

Bei dem implementierten Algorithmus handelt es sich um ein $(\mu+\lambda)-GA$, wobei die Lebensdauer einer Lösung auf κ Generationen beschränkt ist. Wenn somit ein Individuum über κ Generationen in die kommenden Population ausgewählt wurde, so wird die Lösung nicht mehr in der kommenden Population selektiert werden können. Es gilt das Individuum zu selektieren, für welches der Wert der Fitnessfunktion $f(x)$ am geringsten ist.

Umsetzung der Fitnessfunktion

Die umgesetzte Fitnessfunktion $f(x)$ orientiert sich an den Definitionen und der Priorisierungen der Zielfunktionen aus Abschnitt 3.3. Dadurch resultiert eine weitaus stärkere Gewichtung der Projektdauer C_{max} im Vergleich zur Robustheit Ω. Die Fitnessfunktion $f(x)$ ist gemäß dieser Gewichtung über die folgende Formel definiert:

$$f(x) = C_{max}(x) - \frac{\Omega(x)}{100}$$

4.4. Experimente

Die Forschungsfrage und ihre Unterfragen gilt es über quantitative Daten zu beantworten. Hierfür sind im Konzeptüberblick (vgl. Abbildung 3.1) Experimente vorgesehen. Das Klassendiagramm aus Abbildung B.4 zeigt die Umsetzung der einzelnen Experimente auf. Die konkreten Experimente erben vom Interface Experiment, welche eine Methode runExperiment(...) vorsieht. Die Parameter der Methode stellen die

weiterpropagierten Startargumente und die Liste der zu berücksichtigenden Benchmarks dar. Die Experimente werden über die Komponente CommandRunnerComponent gemäß der Startargumente gestartet. Die im Kapitel 5 verwendeten Tabellen werden anhand der aus den Experimenten generierten CSV-Daten erzeugt. Die folgenden Unterabschnitte zeigen die Umsetzungen der Experimente auf.

4.4.1. Vergleich der Lösungsansätze

Für den Vergleich der Lösungsansätze ist die Klasse SolverPerformanceExperiment vorgesehen. Für eine Menge an Benchmarks wird ein Experiment mit den folgenden Variablen parametrisiert:

- Liste von Iterationen
- Liste von Lösungsverfahren
- Robustheitsmessungsfunktion $\Omega(x)$
- Anzahl der Experimente n je Solver/Iteration-Kombination

Jeder Benchmark innerhalb eines Benchmark-Sets wird mit allen vorgesehenen Iterationen und Lösungsverfahren n-Mal durchlaufen. Diese sollen die Makespan C_{max} minimieren und die Robustheit Ω maximieren. Mittels diesem Experiment gilt es quantitativ aufzuzeigen, wie sich die Lösungsverfahren über die Iterationen auswirken und welche sich somit besonders eignen. Das n-malige Durchlaufen von Lösungsverfahren dient dazu, die Tendenz besser bestimmen zu können, da die Verfahren je nach Ausführung unterschiedliche Ergebnisse liefern können. Beim Vergleich der Lösungsverfahren für das (M)RCPSP werden vermehrt die Abweichungen zu den optimalen bzw. der bestbekanntesten Makespan für den Vergleich zur Hand gezogen [vgl. KH98, S. 16 ff.] [vgl. JMR$^+$01, S. 147] [vgl. Har98, S. 10]. Für die n-Mal durchlaufenden Experimente je Lösungsverfahren und Iterationen werden die Mittelwerte aus den Abweichungen zum optimalen Makespan und der Robustheit berechnet. Folglich werden die für ein Lösungsverfahren erreichten Makespan und Robustheitswerte innerhalb einer CSV-Datei im Ordner result gespeichert.

Moderne Computersysteme nutzen vermehrt Multicore-Prozessoren, welche um die Jahrtausendwende über die Zeit immer mehr Kerne aufwiesen [vgl. Har12, S. 12]. Die Implementierung sieht vor, dass die Anzahl der Experimente n innerhalb des Prozessors parallel durchlaufen werden, um so von den Multicore-Prozessoren zu profitieren. Dies vermindert die tatsächliche Durchlaufzeit eines Experimentes.

Für den Vergleich der Verfahren werden ebenfalls die Feasbility- und Optimal-Raten der Algorithmen zur Hand gezogen [vgl. JMR+01, S. 147 ff.] [vgl. Har98, S. 10 f.]. Die Feasbility-Rate gibt an, ob für alle Benchmarks gültige Zeitpläne für das konkrete Lösungsverfahren gefunden wurden. Die Optimal-Rate sagt zudem aus, ob für alle Benchmarks (gemäß der Makespan C_{max}) ein optimaler Zeitplan gefunden wurde. Dadurch, dass in dieser Implementierung mehrere Experimente n für ein Verfahren durchlaufen werden, wird in der Kalkulation der Raten stets nur das beste Experiment berücksichtigt.

4.4.2. Unsicherheiten

Die pro-, prä- und reaktiven Varianten gilt es auf eine Menge an Unsicherheitsszenarien miteinander zu vergleichen. Die Grundlage für alle Varianten stellt die abstrakte Klasse UncertaintyExperiment dar. Diese implementiert das Interface Experiment und die vorgesehene Methode runExperiment(...). Die Parametrisierung entspricht die des Vergleichs der Lösungsansätze aus dem Abschnitt 4.4.1. Je Benchmark werden alle zu berücksichtigenden Lösungsverfahren und Iterationen n-Mal durchlaufen. Zudem sieht die Klasse die abstrakte Methode buildSolution(...) vor, welche von den konkreten Verfahren implementiert werden müssen. Über diese Methode soll die Parametrisierung der Lösungsverfahren (bspw. die Robustheitsoptimierung) für das entsprechende Verfahren erfolgen.

Die Basiszeitpläne gilt es anschließend auf eine Menge an Unsicherheitsszenarien gemäß Abschnitt 3.5 zu evaluieren. Die für jedes Verfahren zu implementierende Methode **int** getUncertaintyExperiments() gibt die Anzahl der Unsicherheitsszenarien je Unsicherheitsmodel zurück, die für jeden Basiszeitplan durchlaufen werden. Aufgrund der Komplexität geschieht dies innerhalb des Prozessors parallel. Über die Methode buildUncertaintySolution() wird das Unsicherheitsszenario auf den Basiszeitplan angewandt. Zudem wird im Fall einer reaktiven Zeitplanerstellung direkt auf Unsicherheiten reagiert.

Die resultierenden Verspätungen verlängern die Makespan C_{max} der Basiszeitpläne. Die Makespan eines Zeitplans nach Anwendung eines Unsicherheitsszenarios wird im Rahmen der Arbeit als C'_{max} definiert und beschreibt die Makespan des eigentlichen bzw. des Verspätungszeitplans. Ziel der einzelnen Verfahren ist es, die resultierende Verspätung zum Basiszeitplan $C^{\Delta}_{max} = C'_{max} - C_{max}$ zu minimieren. Durch eine Vielzahl an Experimenten und Unsicherheitsszenarien entstehen Basis- und Verspätungszeitpläne, für welche die Makespans C_{max}, C'_{max} und Metadaten, wie das angewandte Lösungsverfahren und die Robustheitsmessungsfunkti-

on in einer seperaten CSV-Datei gespeichert werden. Diese werden genutzt um je nach Experimentreihe und Benchmark die Mittelwerte $\overline{C_{max}}$, $\overline{C'_{max}}$, $\overline{C^{\Delta}_{max}}$ je nach Lösungsverfahren zu berechnen.

Insbesondere die reaktive Zeitplanerstellung stellt ein zeitaufwendiges Verfahren dar, da für jede Aktivitätsverspätung eine reaktive Tabu-Suche mit 500 Iterationen ausgeführt wird. Um die Berechnungszeit in der Evaluation zu verringern, wird innerhalb eines PSPLIB-Benchmarkssets nur die erste Instanz eines jeweiligen Parameters betrachtet. Somit werden beispielsweise beim Benchmarkset m1 64 anstelle von 640 Benchmarks durchlaufen. Die folgenden Unterabschnitte erläutern die Realisierungen der einzelnen Verfahren für den Umgang der Unsicherheitsszenarien.

4.4.2.1. Proaktive Zeitplanerstellung

Die Klasse UncertaintyProactiveExperiment stellt das Unsicherheitsexperiment mit der proaktiven Methode dar. Unsicherheiten gilt es hierbei gänzlich zu ignorieren. Die Methode buildSolution(...) erzeugt über die Solver zunächst die Basiszeitpläne, welche die Makespan C_{max} minimieren. Über die Methode buildUncertaintySolution(...) werden auf die Basiszeitpläne die Unsicherheitsszenarien angewandt. Die Anzahl der Unsicherheitsszenarien wurde aufgrund der niedrigen Komplexität auf $m = 50$ festgelegt.

4.4.2.2. Prädiktive Zeitplanerstellung

Bei der prädiktiven Zeitplanerstellung ist die Klasse UncertaintyPredictiveExperiment vorgesehen. Hierbei werden im Vornherein Basiszeitpläne erzeugt, welche neben der Minimierung der Makespan C_{max} die Maximierung der Robustheit Ω als Ziel haben. Über die Methode buildSolution(...) werden somit über die Solver Basiszeitpläne erzeugt, die beide Zielfunktionen gemäß Abschnitt 3.3 optimiert. Die Methode buildUncertaintySolution(...) bleibt gegenüber der proaktiven Zeitplanerstellung aus Abschnitt 4.4.2.1 unverändert. Ebenfalls werden aufgrund der geringen Komplexität $m = 50$ Unsicherheitsszenarien durchlaufen.

Im Abschnitt 2.1.4.1 wurden verschiedene Robustheitsmessungsfunktionen aufgeführt. Zuvor gilt es festzustellen, welche Funktion sich für die Benchmarks-Sets am besten eignet. Um die beste Funktion auszuwählen, wird ein weiteres Experiment, nämlich ein Vergleich der Robustheitsmessungen benötigt. Dieses wird über die Klasse RobustnessExperiment realisiert. Dieses Experiment baut auf dem Schema des UncertaintyExperiment auf, vergleicht aber eine Liste von Robustheitsmessungsfunktionen $\Omega_i^j \, \forall i \in SF, j \in W$ quantitativ auf die Wirksamkeit für $m = 50$ Unsi-

cherheitsszenarien. In der Evaluierung wird die beste Robustheitsmessungsfunktion für das UncertaintyPredictiveExperiment selektiert. Die implementierten Robustheitsmessungsfunktionen sind in Tabelle 4.1 aufgeführt.

Kennung / Klassenname	Slack Function (SF)	Weight (W)	Beschreibung
SF1	SF1	-	Summe von freien Puffern
SF1_W1	SF1	W1	Summe von freien Puffern, gewichtet mit der Anzahl der direkten Nachfolger
SF1_W9	SF1	W9	Summe von freien Puffern, gewichtet mit der Anzahl aller Vorgänger
SF2	SF2	-	Summe von binären Puffern
SF2_W1	SF2	W1	Summe von binären Puffern, gewichtet mit der Anzahl der direkten Nachfolger
SF2_W9	SF2	W9	Summe von binären Puffern, gewichtet mit der Anzahl aller Vorgänger
SF3	SF3	-	Minimum zwischen freien Puffer und Bruchteil der Aktivitätsdauer
SF3_W1	SF3	W1	Minimum zwischen freien Puffer und Bruchteil der Aktivitätsdauer, gewichtet mit der Anzahl der direkten Nachfolger
SF3_W9	SF3	W9	Minimum zwischen freien Puffer und Bruchteil der Aktivitätsdauer, gewichtet mit der Anzahl aller Vorgänger

Tabelle 4.1.: Integrierte Robustheitsmessungfunktionen $\Omega_i^j \, \forall i \in SF, j \in W$

Quelle: Eigene Darstellung

4.4.2.3. Reaktive Zeitplanerstellung

Für die reaktive Zeitplanerstellung ist die Klasse UncertaintyReactiveExperiment vorgesehen. Die Implementierung der Methode buildSolution(...) orientiert sich hierbei an dem proaktiven Äquivalent.

Bei den reaktiven Verfahren werden gemäß Abschnitt 3.2 an den Verspätungszeitpunkten über eine reaktive Tabu-Suche alternative Zeitpläne gefunden. Dieser Algorithmus wurde über die Klasse ReactiveTabuSearchAlgorithm realisiert, welche sich von den anderen Lösungsverfahren dahingehend unterscheidet, dass Zeilpläne gefunden werden, die die Kostenzielfunktion $\mathcal{C}$ aus Abschnitt 2.1.4.2 minimiert. Als initiale Lösung wird bei der reaktiven Tabu-Suche der Basiszeitplan bzw. der aktuelle Zeitplan zur Hand gezogen. Die Größe der reaktiven Tabu-Suche bleibt mit $|TL| = \sqrt{N-2}$ gleich zum Basislösungsverfahren. Da ein Zeitplan bei der reaktiven Tabu-Suche zum Teil ausgeführt wurde, gilt es die durchlaufenden Aktivitäts- und Modusteillisten (List<ActivityMode> fixedActivityModeList) bei dem neuen Zeitplan zu berücksichtigen. Über die Methode SolverHelper.getNeighbourhood(...) werden folglich Zeitpläne in einer Nachbarschaft $s^* \in N(s)$ generiert, welche die bereits durchlaufenden Aktivitäten- und Modi in der Reihenfolge beinhalten.

Aufgrund der hohen Komplexität der reaktiven Verfahren werden 12 anstelle von 50 Unsicherheitsszenarien je Unsicherheitsmodell durchlaufen. Dies ist notwendig, da je nach Unsicherheitsmodell für jedes Unsicherheitsszenario womöglich mehrfach neue Zeitpläne gesucht werden müssen. Durch die Nutzung von mehreren Prozessor-

kernen kann zwar die Erstellung der neuen Zeitpläne parallel durchlaufen werden, dennoch stellt das reaktive Verfahren gegenüber den pro- und prädiktiven Verfahren eine höhere Komplexität im Bezug auf die Berechnungsdauer dar.

Kapitel 5

Evaluation

Das entwickelte MRCPSP-Framework stellt mit seinen Metaheuristiken und Verfahren für den Umgang von Verspätungen die Basis zur Beantwortung der Forschungsfrage und ihrer Unterfragen dar. Dieses Kapitel nutzt die implementierten Lösungsverfahren und Methodiken und vergleicht diese auf Benchmark-Instanzen. Der Versuchsaufbau wird zunächst im Abschnitt 5.1 erläutert. Unabhängig der Verspätungsszenarien werden im Abschnitt 5.2 die Lösungsverfahren auf unterschiedliche Benchmark-Instanzen verglichen, um so die Güte der Verfahren miteinander vergleichen zu können. Die Ergebnisse der proaktiven Methode aus dem Abschnitt 5.3 stellen die Referenz für prädiktive und reaktive Methoden dar. Die Ergebnisse der Robustheitsoptimierung als prädiktive Methode werden im Abschnitt 5.4 und die Ergebnisse der Neuerzeugung von Zeitplänen zum Unsicherheitszeitpunkt als reaktive Methode im Abschnitt 5.5 aufgeführt. Anschließend werden die unterschiedlichen Methoden im Abschnitt 5.6 miteinander verglichen.

5.1. Versuchsaufbau

Der Versuchsaufbau erstreckt sich über die verschiedenen Experimente, welche bereits in der Umsetzung im Abschnitt 4.4 erläutert wurden. Die Auswahl der Benchmark-Instanzsets wurde im Abschnitt 3.4 getroffen. Für die Evaluierung sind die Instanzsets m1, m2, n0, n1 und exklusiv für den Vergleich der Lösungsverfahren j20 der PSPLIB vorgesehen.

Die Ausführung der Experimente wurde auf zwei Computersystemen verteilt ausgeführt. Maßgeblich für die Ausführungsgeschwindigkeit ist die Prozessorleistung. Das erste System stellt ein Desktop-PC mit einem Intel® Core™ i7-8700K mit 6 Kernen und 12 Threads als Prozessor dar. Das zweite System ist ein virtueller Server, welcher als Prozessor einen Intel® Xeon® E5-2680 v3 vorsieht, wobei von den 24 theoretischen Kernen nur 6 zur Verfügung stehen. Für den virtuellen Server lief das Bash-Startskript aus Anhang A im Hintergrundmodus.

Für die Parametrisierung der Experimente sind eine Liste von Iterationen, die Anzahl der Experimente je Solver/Iteration-Kombination und je nachdem eine Robustheitsmessung vorgesehen (vgl. Abschnitt 4.4.1 und 4.4.2).

In der Literatur werden unterschiedliche Listen von Iterationen je Forschungsgegenstand verwendet. Bei dem Lösungsvariantenvergleich sind in der Literatur von [KS97, S. 16] für das (Basis) RCPSP 1 000 und 5 000 Iterationen vorgesehen, bei [JMR+01, S. 148] ist die Menge höher, nämlich 100, 500, 1 000, 5 000, 10 000, 20 000, 50 000 und 100 000, wobei in einigen Instanzsets eine Sättigung zwischen 10 000 - 20 000 Iterationen zu erkennen ist. [WMY14, S. 612] vergleicht die Ergebnisse verschiedener Varianten zwischen 4 000 und 6 000 Iterationen. Im Rahmen dieser Evaluierung ist somit für das Finden der (Basis-)Zeitpläne eine Liste mit 5 00, 1 000, 2 500 und 5 000 Iterationen vorgesehen.

Um die Qualität eines Solvers besser bestimmen zu können, werden diese je nach vorgesehener Iteration mehrfach ausgeführt und die Ergebnisse gemittelt. Die mehrfache Ausführung geschieht im Prozessor parallel (vgl. Abschnitt 4.4.1). In dieser Arbeit richtet sich die Anzahl der Wiederholungen nach der Menge der Rechenkerne. Hierfür wird das zweite System als Referenz verwendet und somit $n = 6$ festgelegt. Dies führt dazu, dass jeder Solver für jede Iteration sechsmal ausgeführt wird.

Eine Robustheitsfunktion Ω wird für den Vergleich der implementierten Lösungsverfahren und der prädiktiven Methoden benötigt. Für den Vergleich der implementierten Lösungsverfahren wird nur die Summe der freien Puffer innerhalb eines Zeitplans betrachtet, welche über die Funktion Ω^{SF1} im Abschnitt 2.1.4.1 definiert ist. Die Auswahl der Robustheitsfunktion Ω für den Unsicherheitsvergleich geschieht über ein seperates Experiment, welches im Abschnitt 5.4 behandelt wird.

Im Anhang D sind die Ergebnisse festgehalten. Durch die Vielzahl der Instanzsets wird für die Evaluierung der Fokus vermehrt auf das Set n1 festgelegt, welches eine moderate Komplexität gegenüber den anderen Instanzsets (m1, m2 und n0) aufweist. Diese gilt es dennoch in der Evaluierung zu berücksichtigen.

5.2. Vergleich der implementierten Lösungsverfahren

Die implementierten Lösungsverfahren wurden über die vollständigen Instanzsets m1, m2, n0, n1 und j20 miteinander verglichen. Tabelle 5.1 zeigt die quantitativen Ergebnisse für das Instanzset n1 auf, während im Anhang D.1 die Ergebnisse der weiteren Instanzsets entnommen werden können.

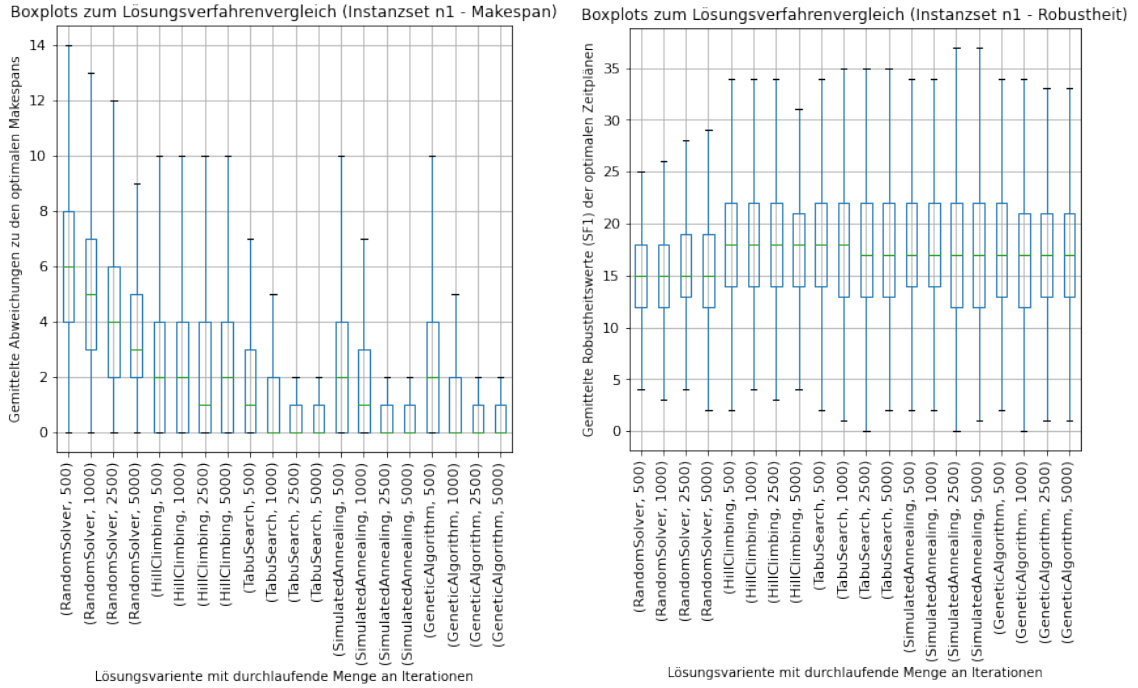
Instance set n1		Dev. Makespan C^{Δ}_{max}		Robustness Ω^{SF1}		Feasible	Optimum
		Mean μ	Std. Dev. σ	Mean μ	Std. Dev. σ	in %	
Solver	Iteration						
RandomSolver	500	6.17	3.47	14.99	4.11	100.00	7.69
	1000	5.30	3.15	15.21	5.06	100.00	10.52
	2500	4.40	2.76	15.55	5.34	100.00	14.91
	5000	3.76	2.57	15.66	5.14	100.00	19.47
HillClimbing	500	2.61	2.97	17.96	6.02	100.00	54.63
	1000	2.48	2.88	17.88	6.01	100.00	56.67
	2500	2.42	2.87	17.96	5.99	100.00	55.26
	5000	2.41	2.78	17.73	6.05	100.00	56.67
TabuSearch	500	1.85	2.37	18.00	6.44	100.00	64.84
	1000	1.37	2.00	17.75	6.64	100.00	75.35
	2500	0.94	1.66	17.54	6.64	100.00	83.05
	5000	0.72	1.43	17.48	6.69	100.00	87.76
SimulatedAnnealing	500	2.60	2.69	17.81	6.38	100.00	50.39
	1000	1.65	2.05	17.81	6.56	100.00	63.42
	2500	0.95	1.46	17.40	6.88	100.00	78.96
	5000	0.59	1.00	17.29	6.96	100.00	87.91
GeneticAlgorithm	500	2.23	2.35	17.62	6.47	100.00	48.04
	1000	1.03	1.39	17.09	6.69	100.00	73.63
	2500	0.49	0.88	17.21	6.72	100.00	87.13
	5000	0.39	0.76	17.30	6.62	100.00	91.21

Tabelle 5.1.: Vergleich der Lösungsverfahren für das Instanzset n1

Quelle: Eigene Darstellung

Beim direkten Vergleich der Lösungsverfahren lässt sich erkennen, dass das zufällige Generieren von Zeitplänen die schlechteste der implementierten Varianten darstellt. Für das Instanzset n1 konnten z. B. nur 19,47 % optimale Lösungen gefunden werden. Dieses Lösungsverfahren eignet sich jedoch für das m1-Instanzset außerordentlich gut, welches mit einer niedrigen Aktivitäts- und Modusanzahl das einfachste Instanzset der Evaluation darstellt und somit auch die geringste Anzahl an Permutationen aufweist. Hierbei konnten die besten Ergebnisse in Bezug auf die gemittelte Abweichung der Makespans, der Robustheit und der Optimum-Rate erreicht werden. Hill Climbing stellt eine Verbesserung zur zufälligen Generierung von Lösungen dar. Dennoch eignet sich das Verfahren im Vergleich auf die implementierten Metaheuristiken weniger, da sowohl die Abweichung zu den besten Makespans als auch die Rate der optimalen Lösungen am geringsten ist. Eine mögliche Begründung liegt darin, dass beim Erreichen von lokalen Optima diese nicht mehr verlassen werden können. Der integrierte Tabu Suche-Algorithmus erreicht schon mit einer geringen Anzahl an Iterationen ($i = 500$) bereits gute Ergebnisse und stellt auch mit höheren Iterationen für sowohl einfachere als auch komplexe Instanzsets solide Ergebnisse. Simulated Annealing liefert über alle Instanzsets hinweg ähnliche Ergebnisse zur Tabu Suche, jedoch mit höheren Abweichungen. Der umgesetzte ge-

netische Algorithmus liefert ab $i = 1000$ Iterationen in den meisten Instanzsets die besten Ergebnisse in Bezug auf die Makespanabweichung und der Optimumsrate. Abbildung 5.1a und 5.1b stellen Boxplots dar, welche die weiteren deskriptiven Statistiken der einzelnen Lösungsvarianten grafisch aufführen.



(a) Makespanabweichungen von C_{max}

(b) Robustheit Ω^{SF1}

Abbildung 5.1.: Boxplot der implementierten Lösungsverfahren für das Instanzset m1 in Bezug auf die gemittelte a) und b)

Quelle: Eigene Darstellungen

5.3. Benchmark-Ergebnisse der proaktiven Methoden

Die implementierten Lösungsverfahren wurden mit der jeweiligen ersten Instanz jedes Parameters der Benchmarksets m1, m2, n0 und n1 verglichen. Hierbei werden $n = 50$ zufällig generierte Unsicherheitsszenarien gemäß Abschnitt 3.5 über das UncertaintyProactiveExperiment durchlaufen und die gemittelten Abweichungen zu den Optima miteinander verglichen. Die selektierte proaktive Methode ignoriert Unsicherheiten. Tabelle 5.2 zeigt die Ergebnisse für das Instanzset n1 auf, während im Anhang D.2 die weiteren Auswertungen für die anderen Instanzsets enthalten sind.

Instance set n1		Mean μ				Std. Dev σ			
	Uncertainty p:	0%	5%	10%	20%	0%	5%	10%	20%
Solver	Iteration								
RandomSolver	500	6.30	7.15	7.91	9.39	3.50	3.82	4.02	4.32

	1000	5.51	6.40	7.18	8.69	3.11	3.47	3.72	4.11
	2500	4.66	5.54	6.34	7.84	2.83	3.32	3.63	3.98
	5000	3.82	4.70	5.52	7.03	2.57	2.89	3.17	3.55
HillClimbing	500	4.05	5.04	5.87	7.43	3.46	3.76	3.93	4.21
	1000	4.10	5.07	5.91	7.42	3.71	3.96	4.16	4.39
	2500	3.90	4.84	5.69	7.22	3.57	3.82	3.99	4.22
	5000	3.77	4.74	5.59	7.11	3.51	3.77	3.97	4.19
TabuSearch	500	1.80	2.77	3.60	5.13	2.29	2.72	2.92	3.27
	1000	1.23	2.24	3.11	4.64	1.86	2.47	2.77	3.12
	2500	0.86	1.85	2.71	4.24	1.57	2.12	2.39	2.75
	5000	0.56	1.54	2.39	3.92	1.14	1.66	1.95	2.35
SimulatedAnnealing	500	2.89	3.78	4.59	6.12	2.50	2.80	3.04	3.39
	1000	1.82	2.78	3.62	5.13	1.99	2.40	2.69	3.01
	2500	0.96	1.90	2.75	4.26	1.37	1.82	2.12	2.52
	5000	0.69	1.71	2.58	4.13	1.05	1.76	2.09	2.53
GeneticAlgorithm	500	2.28	3.15	3.94	5.47	2.30	2.58	2.81	3.14
	1000	1.04	1.92	2.74	4.24	1.35	1.78	2.06	2.43
	2500	0.57	1.48	2.31	3.84	0.90	1.40	1.70	2.10
	5000	0.46	1.40	2.24	3.79	0.81	1.45	1.80	2.22

Tabelle 5.2.: Vergleich der proaktiven Lösungsverfahren auf $m = 50$ unterschiedliche Unsicherheitsszenarien für das Instanzset n1.

Quelle: Eigene Darstellung

Über die Intensität der Unsicherheitsszenarien p lassen sich wesentliche Verspätungen erkennen. Über $p = 0$ lassen sich die Ergebnisse der geplanten Zeitpläne ohne Verspätungen ablesen. Erst mit $p > 0$ können Verspätungen entstehen (vgl. Abschnitt 3.5). Sowohl innerhalb der Tabellen als auch über die Boxplots lassen sich je nach höherer Intensität auch höhere Verspätungen erkennen.

5.4. Benchmark-Ergebnisse der prädiktiven Methoden

Basierend auf dem gleichen Datensatz und Unsicherheitsszenarien vom vorherigen Abschnitt 5.3 werden im Rahmen dieser prädiktiven Methode Zeitpläne gesucht, welche sowohl die Makespan C_{max} minimieren, als auch die Robustheit Ω maximieren. Bereits im Abschnitt 2.1.4.2 wurden unterschiedliche Robustheitsmessungsfunktionen vorgestellt, wovon einige in Abschnitt 4.4.2.3 in das MRPCSP-Framework integriert worden sind.

Zunächst gilt es die Robustheitsmessungsfunktion Ω auszuwählen, welche im Mittel die geringste Verspätung für eine Menge an zufälligen Unsicherheitsszenarien ($n = 50$) je Instanz aufweist. Tabelle 5.3 zeigt die implementierten Robustheitsmes-

sungsfunktionen Ω in Kombination der Tabu-Suche auf. Im Anhang D.3.1 befinden sich für das Instanzset n1 die Resultate für die weiteren Lösungsverfahren. Als Bemessungskriterium werden die gemittelten Verspätungen zu den Basiszeitplänen festgelegt. Es lässt sich erkennen, dass Robustheitsmessungsfunktionen gegenüber keiner Robustheitsoptimierung (No-RM) geringere Projektlaufzeiten aufweisen und somit eine Möglichkeit für den Umgang mit Unsicherheiten darstellt. Insbesondere die Funktionen Ω^{SF2}, Ω^{SF2}_{W1} und Ω^{SF3} stellen bessere Strategien zur Messung der Robustheit innerhalb des MRCPSP dar. Folglich wird die Robustheitoptimierung über die Funktion Ω^{SF2}_{W1} als das zu vergleichende prädiktive Verfahren selektiert.

Instance set n1		Mean μ				Std. Dev σ			
TabuSearch Iteration	Uncertainty p: $\Omega(x)$	0%	5%	10%	20%	0%	5%	10%	20%
500	No-RM	0.00	1.01	1.87	3.46	0.00	1.45	1.81	2.28
	SF1	0.00	0.90	1.71	3.21	0.00	1.31	1.67	2.21
	SF1_W1	0.00	0.89	1.72	3.18	0.00	1.27	1.68	2.11
	SF1_W9	0.00	0.94	1.71	3.27	0.00	1.24	1.58	2.10
	SF2	0.00	0.86	1.65	3.14	0.00	1.21	1.63	2.08
	SF2_W1	0.00	0.86	1.63	3.10	0.00	1.22	1.55	2.01
	SF2_W9	0.00	0.89	1.70	3.21	0.00	1.36	1.78	2.24
	SF3	0.00	0.87	1.67	3.16	0.00	1.19	1.56	2.08
	SF3_W1	0.00	0.86	1.65	3.15	0.00	1.18	1.52	1.96
	SF3_W9	0.00	0.88	1.72	3.24	0.00	1.31	1.77	2.28
1000	No-RM	0.00	0.98	1.84	3.35	0.00	1.38	1.75	2.23
	SF1	0.00	0.93	1.74	3.26	0.00	1.28	1.68	2.13
	SF1_W1	0.00	0.92	1.77	3.29	0.00	1.33	1.79	2.25
	SF1_W9	0.00	0.92	1.76	3.26	0.00	1.23	1.65	2.13
	SF2	0.00	0.89	1.70	3.20	0.00	1.34	1.68	2.16
	SF2_W1	0.00	0.86	1.64	3.10	0.00	1.22	1.54	1.99
	SF2_W9	0.00	0.86	1.67	3.13	0.00	1.20	1.60	1.99
	SF3	0.00	0.88	1.71	3.24	0.00	1.24	1.69	2.11
	SF3_W1	0.00	0.89	1.68	3.22	0.00	1.25	1.63	2.11
	SF3_W9	0.00	0.89	1.71	3.20	0.00	1.27	1.70	2.10
2500	No-RM	0.00	1.02	1.91	3.46	0.00	1.38	1.79	2.26
	SF1	0.00	0.97	1.81	3.32	0.00	1.35	1.77	2.20
	SF1_W1	0.00	0.93	1.71	3.22	0.00	1.27	1.60	2.05
	SF1_W9	0.00	0.96	1.79	3.33	0.00	1.39	1.76	2.29
	SF2	0.00	0.85	1.64	3.17	0.00	1.18	1.58	2.07
	SF2_W1	0.00	0.85	1.64	3.16	0.00	1.13	1.53	1.96
	SF2_W9	0.00	0.88	1.69	3.19	0.00	1.21	1.64	2.07
	SF3	0.00	0.91	1.71	3.23	0.00	1.39	1.79	2.23
	SF3_W1	0.00	0.90	1.71	3.24	0.00	1.28	1.67	2.15
	SF3_W9	0.00	0.94	1.76	3.29	0.00	1.40	1.75	2.24
5000	No-RM	0.00	0.99	1.86	3.39	0.00	1.20	1.54	1.94

SF1	0.00	0.95	1.78	3.29	0.00	1.31	1.67	2.13
SF1_W1	0.00	0.96	1.77	3.29	0.00	1.35	1.74	2.18
SF1_W9	0.00	0.98	1.83	3.39	0.00	1.39	1.81	2.29
SF2	0.00	0.85	1.63	3.12	0.00	1.14	1.52	2.02
SF2_W1	0.00	0.86	1.64	3.14	0.00	1.15	1.50	1.97
SF2_W9	0.00	0.92	1.71	3.21	0.00	1.33	1.67	2.11
SF3	0.00	0.90	1.74	3.23	0.00	1.25	1.62	2.03
SF3_W1	0.00	0.91	1.73	3.23	0.00	1.28	1.64	2.06
SF3_W9	0.00	0.97	1.81	3.32	0.00	1.44	1.85	2.25

Tabelle 5.3.: Verspätungswerte der Robustheitfunktionen $\Omega(x)$ angewendet auf die Tabu Suche für das Instanzset n1 mit $n = 50$ Unsicherheitsszenarien.

Quelle: Eigene Darstellung

Für alle zu untersuchenden Benchmark Sets gilt es die Robustheit über die ausgewählte Robustheitsmessungsfunktion Ω_{W1}^{SF2} zu maximieren und die Makespan C_{max} zu minimieren. Dies geschieht über den UncertaintyPredictiveExperiment, welcher zudem nun Basispläne über die Robustheit sekundär auswählt. Tabelle 5.4 zeigt die Ergebnisse des prädiktiven Verfahrens für das Instanzset n1 auf. Im Anhang D.3 befinden sich die Ergebnisse für die weiteren Instanzsets.

Instance set n1		Mean μ				Std. Dev σ			
	Uncertainty p:	0%	5%	10%	20%	0%	5%	10%	20%
Solver	Iteration								
RandomSolver	500	6.16	6.97	7.74	9.21	3.28	3.55	3.78	4.16
	1000	5.43	6.27	7.04	8.56	3.16	3.53	3.78	4.17
	2500	4.47	5.28	6.09	7.54	2.74	3.04	3.27	3.63
	5000	3.90	4.70	5.48	6.95	2.60	2.89	3.10	3.46
HillClimbing	500	2.56	3.41	4.20	5.70	2.94	3.24	3.46	3.84
	1000	2.40	3.22	4.00	5.46	2.63	2.95	3.21	3.56
	2500	2.41	3.25	4.03	5.52	2.76	3.04	3.22	3.56
	5000	2.42	3.26	4.05	5.55	2.93	3.21	3.43	3.80
TabuSearch	500	1.60	2.47	3.27	4.76	2.17	2.52	2.79	3.10
	1000	1.26	2.12	2.90	4.38	1.96	2.30	2.53	2.91
	2500	0.86	1.70	2.49	4.00	1.55	1.95	2.23	2.62
	5000	0.61	1.46	2.26	3.75	1.33	1.71	1.97	2.34
SimulatedAnnealing	500	2.51	3.38	4.20	5.71	2.65	3.08	3.36	3.79
	1000	1.74	2.60	3.40	4.87	2.25	2.61	2.86	3.17
	2500	0.92	1.79	2.55	4.06	1.29	1.82	2.13	2.54
	5000	0.62	1.48	2.28	3.78	1.01	1.60	1.90	2.32
GeneticAlgorithm	500	2.17	2.99	3.77	5.25	2.15	2.42	2.61	2.94
	1000	1.09	1.92	2.70	4.18	1.37	1.76	2.03	2.42
	2500	0.57	1.42	2.20	3.67	0.90	1.43	1.74	2.16
	5000	0.43	1.27	2.05	3.54	0.80	1.35	1.69	2.10

Tabelle 5.4.: Vergleich der prädiktiven Lösungsverfahren auf $m = 50$ unterschiedliche Unsicherheitsszenarien für das Instanzset n1.

Quelle: Eigene Darstellung

5.5. Benchmark-Ergebnisse der reaktiven Methoden

Auf der identischen Datenbasis zu Abschnitt 5.3 und 5.4 werden in diesem Abschnitt die Ergebnisse der reaktiven Methode vorgestellt. Hierbei werden zunächst Basiszeitpläne gefunden, welche die Makespan C_{max} minimieren. Bei einem Verspätungszeitpunkt wird anschließend ein Folgezeitplan über eine Tabu Search gesucht, welcher die Kostenfunktion $\mathcal{C}$ aus Abschnitt 2.1.4.2 minimiert. Der Algorithmus nutzt als initiale Lösung den aktuellen Zeitplan und durchläuft 500 Iterationen. Die Ergebnisse der Methode für das Instanzset n1 sind in der Tabelle 5.5 visualisiert. Weitere Ergebnisse sind aus dem Anhang D.4 zu entnehmen.

Instance set n1		Mean μ				Std. Dev σ			
	Uncertainty p:	0%	5%	10%	20%	0%	5%	10%	20%
Solver	Iteration								
RandomSolver	500	6.27	6.61	6.97	7.74	3.51	3.56	3.63	3.88
	1000	5.64	6.03	6.40	7.16	3.32	3.40	3.48	3.52
	2500	4.69	5.08	5.52	6.37	2.95	3.00	3.08	3.22
	5000	3.73	4.21	4.66	5.65	2.45	2.53	2.68	2.85
HillClimbing	500	3.82	4.39	4.93	5.98	3.51	3.47	3.44	3.41
	1000	3.84	4.44	4.93	6.04	3.38	3.41	3.41	3.39
	2500	3.64	4.21	4.76	5.84	3.13	3.15	3.18	3.18
	5000	3.94	4.54	5.10	6.05	4.06	4.09	4.11	3.96
TabuSearch	500	1.77	2.45	3.17	4.44	2.20	2.29	2.39	2.65
	1000	1.21	1.96	2.68	4.00	1.81	2.01	2.17	2.28
	2500	0.98	1.76	2.53	3.94	1.93	2.13	2.26	2.50
	5000	0.50	1.36	2.10	3.64	0.99	1.40	1.65	2.04
SimulatedAnnealing	500	2.91	3.51	4.02	5.21	2.55	2.59	2.68	2.78
	1000	1.81	2.52	3.17	4.42	2.14	2.33	2.44	2.70
	2500	1.05	1.81	2.54	3.86	1.41	1.73	1.90	2.20
	5000	0.67	1.51	2.27	3.67	1.05	1.48	1.76	2.02
GeneticAlgorithm	500	2.17	2.78	3.36	4.52	2.12	2.11	2.20	2.26
	1000	1.15	1.85	2.50	3.90	1.42	1.58	1.71	2.06
	2500	0.55	1.39	2.16	3.56	0.89	1.38	1.62	2.06
	5000	0.49	1.32	2.10	3.53	0.87	1.39	1.62	2.02

Tabelle 5.5.: Vergleich der reaktiven Lösungsverfahren auf $m = 50$ unterschiedliche Unsicherheitsszenarien für das Instanzset n1.

Quelle: Eigene Darstellung

5.6. Vergleich der proaktiven, prädiktiven und reaktiven Methoden

Die unterschiedlichen Verfahren zum Umgang mit Unsicherheiten gilt es in diesem Abschnitt miteinander zu vergleichen. Bereits in den vorherigen Abschnitten wurden Ergebnisse der Verfahren für verschiedene Unsicherheitsszenarien aufgeführt. Für den Vergleich dienen die Ergebnisse der proaktiven Evaluation aus Abschnitt 5.3 als Referenz, da Verspätungen ignoriert werden. Im Hinblick auf die Forschungsfrage aus Abschnitt 1.2 gilt es zu überprüfen, wie sich der prädiktive und reaktive Ansatz für das MRCPSP auf Basis von metaheuristischen Algorithmen bei der Erstellung von Zeitplänen in Bezug auf die Projektdauer bei Verspätungen verhält.

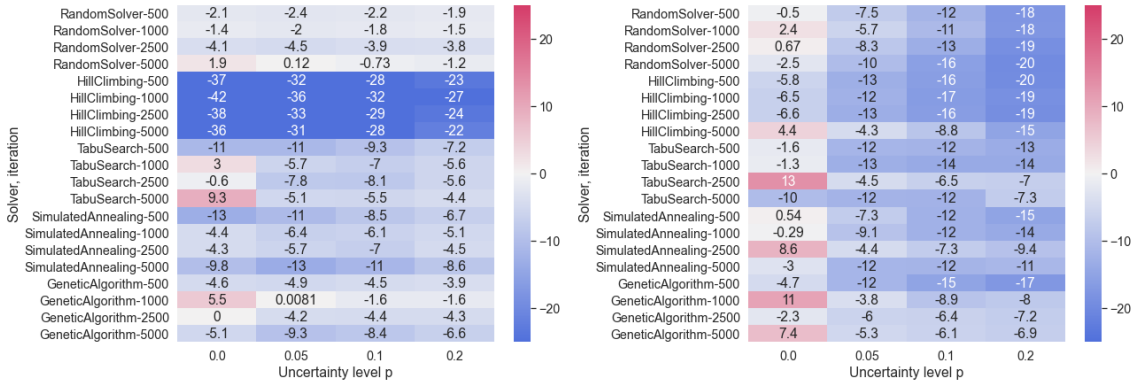
Instance set n1		Proactive Mean μ				Predictive Mean μ				Reactive Mean μ			
	Uncertainty p:	0.00	0.05	0.10	0.20	0.00	0.05	0.10	0.20	0.00	0.05	0.10	0.20
Solver	Iteration												
RandomSolver	500	6.30	7.15	7.91	9.39	6.16	6.97	7.74	9.21	6.27	6.61	6.97	7.74
	1000	5.51	6.40	7.18	8.69	5.43	6.27	7.04	8.56	5.64	6.03	6.40	7.16
	2500	4.66	5.54	6.34	7.84	4.47	5.28	6.09	7.54	4.69	5.08	5.52	6.37
	5000	3.82	4.70	5.52	7.03	3.90	4.70	5.48	6.95	3.73	4.21	4.66	5.65
HillClimbing	500	4.05	5.04	5.87	7.43	2.56	3.41	4.20	5.70	3.82	4.39	4.93	5.98
	1000	4.10	5.07	5.91	7.42	2.40	3.22	4.00	5.46	3.84	4.44	4.93	6.04
	2500	3.90	4.84	5.69	7.22	2.41	3.25	4.03	5.52	3.64	4.21	4.76	5.84
	5000	3.77	4.74	5.59	7.11	2.42	3.26	4.05	5.55	3.94	4.54	5.10	6.05
TabuSearch	500	1.80	2.77	3.60	5.13	1.60	2.47	3.27	4.76	1.77	2.45	3.17	4.44
	1000	1.23	2.24	3.11	4.64	1.26	2.12	2.90	4.38	1.21	1.96	2.68	4.00
	2500	0.86	1.85	2.71	4.24	0.86	1.70	2.49	4.00	0.98	1.76	2.53	3.94
	5000	0.56	1.54	2.39	3.92	0.61	1.46	2.26	3.75	0.50	1.36	2.10	3.64
SimulatedAnnealing	500	2.89	3.78	4.59	6.12	2.51	3.38	4.20	5.71	2.91	3.51	4.02	5.21
	1000	1.82	2.78	3.62	5.13	1.74	2.60	3.40	4.87	1.81	2.52	3.17	4.42
	2500	0.96	1.90	2.75	4.26	0.92	1.79	2.55	4.06	1.05	1.81	2.54	3.86
	5000	0.69	1.71	2.58	4.13	0.62	1.48	2.28	3.78	0.67	1.51	2.27	3.67
GeneticAlgorithm	500	2.28	3.15	3.94	5.47	2.17	2.99	3.77	5.25	2.17	2.78	3.36	4.52
	1000	1.04	1.92	2.74	4.24	1.09	1.92	2.70	4.18	1.15	1.85	2.50	3.90
	2500	0.57	1.48	2.31	3.84	0.57	1.42	2.20	3.67	0.55	1.39	2.16	3.56
	5000	0.46	1.40	2.24	3.79	0.43	1.27	2.05	3.54	0.49	1.32	2.10	3.53

Tabelle 5.6.: Vergleich der Ergebnisse der pro-, prä- und reaktiven Verfahren für das Instanzset n1

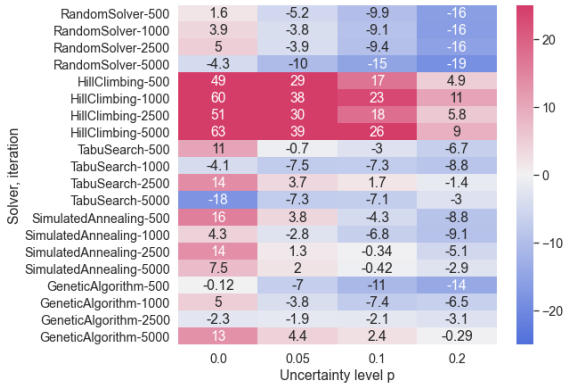
Quelle: Eigene Darstellung

Tabelle 5.6 fasst die Mittelwerte der Abweichungen zu den Optima der einzelnen Verfahren aus den vorherigen Abschnitten für das Instanzset n1 zusammen. Im Anhang D.5 befinden sich die Tabellen für die weiteren Benchmarks.

Diese Mittelwerte gilt es anschließend miteinander prozentual zu vergleichen. In Abbildung 5.2 werden in einer Heatmap-Darstellung die einzelnen Verfahren miteinander verglichen. Positive Werte stellen prozentuale Zunahmen und negative Werte prozentuale Abnahmen dar.



(a) Prädiktive gegenüber proaktive Methode (b) Reaktive gegenüber proaktive Methode



(c) Reaktive gegenüber prädiktive Methode

Abbildung 5.2.: Heatmaps der prozentualen Abweichungen

Quelle: Eigene Darstellungen

Aus den Daten lässt sich erkennen, dass beim Ignorieren von Verspätungen die Abweichungen zu den Optima im Schnitt höher gegenüber der prä- und reaktiven Methode sind. Durch die Anzahl der durchlaufenden Experimente und Unsicherheitsszenarien kann bewiesen werden, dass die selektierten prädiktiven und reaktiven Verfahren einen positiven Einfluss auf die Minimierung der Verspätungen haben.

Innerhalb der Heatmaps lassen sich bei den Verfahren Trends über die Steigung der Unsicherheitslevel p erkennen. Beim prädiktiven Ansatz über die Robustheitsoptimierung ist der beste Effekt bei $p = 0.05$ zu erkennen. Über die Erhöhung des Unsicherheitslevels sinken jedoch die prozentualen Unterschiede zum proaktiven Ansatz. Der reaktive Ansatz über das Reparieren der Zeitpläne zum Unsicherheitszeitpunkt hingegen verbessert sich über die Steigung des Unsicherheitslevels p, insbesondere bei einer geringen Anzahl an Iterationen bei den Basiszeitplänen. Insbesondere bei einem hohen Unsicherheitslevel ($p = 0.2$) zeigt das Verfahren deutlich höhere Unterschiede gegenüber dem proaktiven Ansatz und moderate Unterschiede gegenüber dem prädiktiven Ansatz auf. Für das MRCPSP eignen sich prädiktive Verfahren

besser für niedrigere Störungen, während reaktive Verfahren besser für komplexere Störungen geeignet sind. Dies wiederum bestätigt das, was bereits für das Basisproblem RCPSP von [vgl. BKF12, S. 405 f.] über den Vergleich verschiedener Arbeiten herausgearbeitet wurde.

Für den direkten Vergleich von prädiktiven Verfahren gegenüber reaktiven Verfahren lassen sich jedoch Probleme feststellen, welche auf die Berechnungsdauer bezogen sind. Während prädiktive Verfahren eine feste Anzahl an Iterationen durchlaufen, werden beim reaktiven Ansatz Zeitpläne zum Unsicherheitszeitpunkt repariert, indem neue Zeitpläne gesucht werden. Hierfür wurde im Rahmen dieser Arbeit eine angepasste Tabu Search verwendet, welche jeweils 500 Iterationen durchläuft. Dies führt dazu, dass ein weitaus höherer zeitlicher Aufwand für den reaktiven Umgang betrieben wird als gegenüber dem prädiktiven Ansatz. Basiszeitpläne, die mit einer geringen Anzahl an Iterationen gefunden wurden, profitieren von der reparierenden Tabu Suche, sodass bessere Zeitpläne gefunden werden, die bereits die Aktivitätsstörung zum Unsicherheitszeitpunkt in Betracht ziehen. Dies führt dazu, dass mit Steigung des Unsicherheitslevels die prozentualen Abweichungen bei diesen Basiszeitplänen ebenfalls steigen. Ebenfalls lassen sich bei den schwächeren Lösungsverfahren (Random Solver und Hill Climbing) in Tabelle 5.6 deutlich geringere Differenzen zu den Basiszeitplänen feststellen. Diese Effekte gilt es bei der Interpretierung der Ergebnisse zu berücksichtigen und machen den direkten Vergleich schwieriger.

Bei den nachbarschaftsbasierten Algorithmen (Hill Climbing, Tabu Search und Simulated Annealing) lässt sich zudem ein weiterer Nebeneffekt innerhalb der Basiszeitpläne beim prädiktiven Ansatz erkennen. Es werden im Schnitt bessere Ergebnisse erzielt, wenn ein Algorithmus die Robustheit Ω mit berücksichtigt wird. Dies ist insbesondere in den Heatmaps aus den Abbildungen 5.2a, D.1a, D.2a zu erkennen. Insbesondere Hill Climbing profitiert von der Robustheitsoptimierung, sodass signifikant bessere Basiszeitpläne erzeugt wurden, welche weiterhin im Vergleich zu den Basiszeitplänen der Tabu Suche schlechter ausfallen.

Des Weiteren lässt sich in Abbildung 5.6 erkennen, dass die Güte der Metaheuristiken auch nach Anwendung der Unsicherheitsszenarien für alle Verfahren wesentlich für die Ergebnisse sind. Ein direkter Vergleich ist somit schwierig. Im Abschnitt 5.2 zeigte sich insbesondere der GA als statistisch bestes der implementierten Lösungsverfahren bei der Findung von Basiszeitplänen für eine Vielzahl an Benchmark Sets. Dies und das Muster aller Lösungsvarianten ist auch nach Anwendung der Unsicherheitsszenarien zu erkennen, jedoch etwas abgeschwächter als zwischen den

Basiszeitplänen. Die Wahl der Lösungsvarianten und die Anzahl der zu durchlaufenden Iterationen gilt es ebenfalls zu optimieren, um so die bestmöglichen Zeitpläne auch mit Verspätungen zu erhalten.

Die Intensität der Verfahren für den Umgang von Unsicherheiten hängt nicht zuletzt vom Benchmark Set ab. Während beim Benchmark Set m1 die prozentualen Abweichungen eher gering sind (vgl. Abbildung D.1), sind diese bei den komplexeren Sets m2 (vgl. Abbildung D.2), n1 (vgl. Abbildung 5.2) und n0 (vgl. Abbildung D.3) stärker.

Kapitel 6

Zusammenfassung und Ausblick

Der abschließende Teil dieser Masterarbeit besteht aus einem Fazit und einem Ausblick. Im Abschnitt 6.1 werden die Kernthemen der Masterarbeit zusammengefasst und die wichtigsten Ergebnisse vorgestellt um die Forschungsfrage und dessen Unterfragen zu beantworten. Eine Eingliederung innerhalb des Forschungsstands wird mit einem Ausblick im Abschnitt 6.2 aufgeführt.

6.1. Fazit

Das Thema der Masterarbeit basiert auf dem MRCPSP, welches als NP-vollständig klassifiziert wurde. In dieser Arbeit wurde für das Problem ein Framework konzipiert und implementiert, welches Zeitpläne anhand von Aktivitäts- und Moduslisten erzeugt. Diese wiederum werden über eine Vielzahl an Lösungsvarianten erzeugt, um so die Zeitpläne mit der kürzesten Projektdauer C_{max} zu finden. Insbesondere die implementierten Metaheuristiken Tabu Search, Simulated Annealing und Genetic Algorithm dienen dazu, Lösungen zu finden, welche das globale Optimum annähern. Initiale Zeitpläne wurden über das S-SGS und etabilierte heuristische Aktivitäts- und Selektionsregeln erzeugt. Die im Forschungsfeld anerkannte PSPLIB beinhaltet unterschiedliche Benchmark Sets für das MRCPSP, die über das Framework geladen werden können.

Unsicherheiten, wie Aktivitätsstörungen führen zu Verspätungen bei den Projektplänen. Unsicherheitsszenarien bestehen aus Mengen von Aktivitätsstörungen, die mit einer Binomialverteilung erzeugt werden konnten. Zur Minimierung der Verspätungen wurden pro-, prä- und reaktive Verfahren konzipiert und implementiert. Um die Forschungsfrage zu beantworten, galt es diese Verfahren auf Benchmark Sets und eine Menge an zufallsgenerierten Unsicherheitsszenarien zu evaluieren. Das proaktive Verfahren und das Ignorieren von Verspätungen, dient hierbei als Referenzverfahren. Bei dem prädiktiven Verfahren wurde bereits im Suchprozess über die Metaheuristiken eine weitere Zielfunktion, nämlich die Robustheit Ω berücksichtigt,

welche es zu maximieren gilt. Hierfür wurden zunächst unterschiedliche etablierte Robustheitsmessungsfunktionen selektiert und implementiert. Diese galt es anschließend auf Benchmark Sets und Unsicherheitsszenarien quantitativ zu vergleichen. Als reaktives Verfahren wurde das Reparieren von Zeitplänen zu den Unsicherheitszeitpunkten ausgewählt. Für die Reparatur wurde eine erweiterte Tabu Suche entwickelt, welche mit einer festen Anzahl an Iterationen zum Zeitpunkt reagiert. Diese minimiert die Kostenfunktion $\mathcal{C}$. Somit galt es zur Laufzeit Zeitpläne zu finden, die sich nah am Basiszeitplan orientieren und trotzdem die Verspätung zur geplanten C_{max} minimiert.

Die Forschungsfrage befasste sich mit der Evaluierung der Auswirkungen von prädiktiven und reaktiven Verfahren auf die Projektdauer bei Verspätungen im MRCPSP. Mithilfe eines quantitativen Vergleiches zwischen den implementierten Repräsentanten beider Verfahren gilt es die Forschungsfrage zu beantworten. Die Auswertung zeigte auf, dass sich beide Verfahren für die Verminderung der Projektverspätungen eignen. Der Grad des Unsicherheitslevels ist maßgeblich für die konkrete Auswahl des Verfahrens. Bei einem geringen Unsicherheitslevel zeigte im Schnitt das prädiktive Verfahren über die Robustheitsoptimierung eine bessere Wirkung als das reaktive Verfahren. Umgekehrt zeigte das reaktive Verfahren bei einem hohen Unsicherheitslevel eine bessere Wirkung, als das prädiktive Verfahren. Ein direkter Vergleich ist dennoch schwierig, da beim reaktiven Ansatz die Berechnungsdauer signifikant höher als beim prädiktiven Ansatz ist. Zudem eignen sich reaktive Verfahren bei Basiszeitplänen, die mit einer geringen Anzahl an Iterationen gefunden wurden. Hierbei pendeln sich über die Reparaturvorgänge womöglich bessere Zeitpläne ein und profitieren somit stark von der Güte des Reparaturalgorithmus. Reaktive Verfahren können in der Praxis insbesondere für flexible Projekte genutzt werden, während prädiktive Verfahren die Pläne über den Puffer aufrecht erhalten. Unabhängig der prädiktiven und reaktiven Verfahren stellen Metaheuristiken wesentliche Techniken zur Findung von adäquaten Basiszeitplänen dar. Es zeigte sich zudem, dass die Unterschiede zwischen den Lösungsverfahren und durchlaufenden Iterationen sowohl bei den Basiszeitplänen als auch bei den verspäteten Zeitplänen in der Relation betrachtet weiterhin vorhanden sind.

6.2. Ausblick

Weiterführende Arbeiten können auf den Resultaten dieser Masterthesis aufgebaut werden. Diese Arbeit knüpft an die Erkenntnisse von [BKF12] an, in welcher für das Basisproblem RCPSP ein Vergleich von prädiktiven und reaktiven Verfahren anhand verschiedener Publikationen vollzogen wurde. Diese Masterthesis betrachtete

das MRCPSP, eine Generalisierung des RCPSP, und verglich quantitativ jeweils ein prädiktives und reaktives Verfahren innerhalb eines eigens implementierten Frameworks mit verschiedenen Metaheuristiken und Iterationen auf verschiedene Benchmark Sets und liefert quantitative Daten und Ergebnisse.

Die Erweiterbarkeit des Frameworks war im Mittelpunkt der Konzeption und Implementierung. Über die Interfaces können weitere (Meta-)Heuristiken, Metriken, Experimente oder Benchmark Loader implementiert werden.

Als Lösungsverfahren wurden drei Metaheuristiken selektiert, konzipiert und implementiert, nämlich Tabu Search, Simulated Annealing und Genetic Algorithm. Der aktuelle Trend geht neben der Entwicklung weiterer Metaheuristiken in die Richtung von hybriden Metaheuristiken. Der Einsatz von derartigen Metaheuristiken für das MRCPSP innerhalb des Frameworks bietet weiterhin Potenzial und führt möglicherweise zu besseren Ergebnissen. Auch die Nutzung von parallelen oder verteilten Systemen innerhalb der Lösungsverfahren könnte in Zukunft mehr von Relevanz sein.

Bei der Robustheitsoptimierung wurde eine Vielzahl an etablierten Robustheitsmessungsfunktionen ausgewählt, implementiert und auf eine Menge von Unsicherheitsszenarien auf bestimmte Benchmark Sets miteinander verglichen. In dieser Thesis wurde somit die Robustheitsmessungsfunktion Ω_{W1}^{SF2} selektiert, welche den binären Puffer aller Aktivitäten miteinander addiert und mit der Anzahl der Nachfolger gewichtet ist. Weitere Arbeiten könnten sich auf die Selektion und Entwicklung weiterer Robustheitsmessungsfunktionen konzentrieren.

Im Rahmen des reaktiven Verfahrens wurden Basiszeitpläne ohne Robustheitsoptimierung zur Hand gezogen. Gemäß [vgl. BKF12, S. 404 f.] handelt es sich somit bei der umgesetzten Variante um ein proaktiv-reaktives Verfahren. Die Auswirkung von prädiktiv-reaktiven Verfahren, welche die Robustheitsmaximierung innerhalb der Basiszeitpläne vorsieht, kann interessante Ergebnisse liefern, welche in Folgearbeiten evaluiert werden können.

Zu jedem Unsicherheitszeitpunkt wurde mithilfe einer erweiterten Tabu Suche mit 500 Iterationen der kaputte Zeitplan über die Minimierung der Kostenfunktion $\mathcal{C}$ repariert. Neben Anpassungen an dem Reperaturalgorithmus können auch weitere Verfahren selektiert oder entworfen werden, um die hohe Berechnungsdauer zum Unsicherheitszeitpunkt zu senken und um das globale Minimum der Kostenfunktion besser anzunähern.

Diese Arbeit befasste sich mit Aktivitätsstörungen als einzige Unsicherheitsquelle. Folgearbeiten könnten sich auf weitere Unsicherheitsquellen, wie beispielsweise (nicht-)erneuerbare Ressourcenstörungen beziehen.

Die Forschungsfrage und ihre Unterfragen konnten auf quantitativer Basis beantwortet werden. Mögliche Folgearbeiten könnten den Fokus auf die qualitative Forschung legen. Die erhobenen Daten dieser Arbeit können zur Hand gezogen werden, um festzustellen, wie die Projektdauer sich über die Unsicherheitsszenarien bei Anwendung der Verfahren konkret verändert. Auch die Relevanz der Metadaten zum Projekt, wie die Anzahl der Aktivitäten, Modi, der (nicht)-erneuerbaren Ressourcen, können in einer qualitativen Erhebung genauer betrachtet werden.

Diese Masterthesis bezieht sich mit dem MRCPSP auf ein mathematisches Optimierungsproblem, welches in der Praxis, insbesondere bei der Projektplanung, Relevanz findet. Die Transformation zu Projekten in der Praxis und der einhergehenden Evaluierung von prädiktiven und reaktiven Verfahren stellt ebenfalls Forschungsbedarf dar, da sich möglicherweise weitere Anforderungen und Faktoren identifizieren lassen. In Kombination mit einer qualitativen Erhebung können hierbei neue Erkenntnisse entstehen.

Literaturverzeichnis

[AB21] AT&T and Bell Labs. Graphviz, April 2021. URL: https://graphviz.org/.

[AFH05] M.A. Al-Fawzan and M. Haouari. A bi-objective model for robust resource-constrained project scheduling. *International Journal of Production Economics*, 96(2):175–187, May 2005. URL: https://linkinghub.elsevier.com/retrieve/pii/S0925527304001458, doi:10.1016/j.ijpe.2004.04.002.

[AMR03] J. Alcaraz, C. Maroto, and R. Ruiz. Solving the Multi-Mode Resource-Constrained Project Scheduling Problem with genetic algorithms. *Journal of the Operational Research Society*, 54(6):614–626, June 2003. doi:10.1057/palgrave.jors.2601563.

[ASA06] B. Abbasi, S. Shadrokh, and J. Arkat. Bi-objective resource-constrained project scheduling with robustness and makespan criteria. *Applied Mathematics and Computation*, 180(1):146–152, September 2006. URL: https://linkinghub.elsevier.com/retrieve/pii/S0096300305011057, doi:10.1016/j.amc.2005.11.160.

[Aug19] S. Augsten. Was ist das Spring Framework?, July 2019. URL: https://www.dev-insider.de/was-ist-das-spring-framework-a-829846/.

[BKF12] M. Brčič, D. Kalpić, and K. Fertalj. Resource Constrained Project Scheduling under Uncertainty: A Survey. *Central European Conference on Information and Intelligent Systems*, 2012. URL: https://www.semanticscholar.org/paper/Resource-Constrained-Project-Scheduling-under-A-Br%C4%8Di%C4%8D-Kalpi%C4%87/d94dc395bd160c97495a8cbb3f100660b091bbc6.

[Bur99] R. Burke. *Project management: planning and control techniques*. J. Wiley, Chichester, England ; New York, 3rd ed edition, 1999.

[CLP14] A. Chen, Y. Liang, and J. Padilla. An Entropy-Based Upper Bound Methodology for Robust Predictive Multi-Mode RCPSP Schedules.

Entropy, 16(9):5032–5067, September 2014. URL: http://www.mdpi.com/1099-4300/16/9/5032, doi:10.3390/e16095032.

[DDH08] F. Deblaere, E. Demeulemeester, and W. Herroelen. Exact and Heuristic Reactive Planning Procedures for Multimode Resource-Constrained Projects. SSRN Scholarly Paper ID 1288546, Social Science Research Network, Rochester, NY, August 2008. URL: https://papers.ssrn.com/abstract=1288546.

[DDH11] F. Deblaere, E. Demeulemeester, and W. Herroelen. Reactive scheduling in the multi-mode RCPSP. *Computers & Operations Research*, 38(1):63–74, January 2011. URL: https://www.sciencedirect.com/science/article/pii/S030505481000002X, doi:10.1016/j.cor.2010.01.001.

[EUPEO58] Dwight D Eisenhower, United States, United States President (1953-1961 : Eisenhower), and Office of the Federal Register. *Dwight D. Eisenhower: 1957 : containing the public messages, speeches, and statements of the president, January 1 to December 31, 1957.* Office of the Federal Register, National Archives and Records Service, General Services Administration : For sale by the Supt. of Docs., U.S. G.P.O., Washington, 1958. OCLC: 68945729. URL: http://name.umdl.umich.edu/4728417.

[Gil21] D. Gilbert. JFreeChart, February 2021. URL: https://www.jfree.org/jfreechart/.

[GP19] M. Gendreau and J. Potvin, editors. *Handbook of Metaheuristics*. Number 272 in International Series in Operations Research & Management Science. Springer International Publishing : Imprint: Springer, Cham, 3rd ed. 2019 edition, 2019.

[Har98] S. Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL)*, 45(7):733–750, 1998. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291520-6750%28199810%2945%3A7%3C733%3A%3AAID-NAV5%3E3.0.CO%3B2-C, doi:10.1002/(SICI)1520-6750(199810)45:7<733::AID-NAV5>3.0.CO;2-C.

[Har12] W. Harrod. A journey to exascale computing. In *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, pages 1702–1730, November 2012. doi:10.1109/SC.Companion.2012.345.

[HASB17] A. Hosseinabadi, F. Alavipour, S. Shamshirbnd, and Valentina E. Balas. A Novel Meta-Heuristic Combinatory Method for Solving Capacitated Vehicle Location-Routing Problem with Hard Time Windows. In Valentina E. Balas, Lakhmi C. Jain, and Xiangmo Zhao, editors, *Information Technology and Intelligent Transportation Systems*, Advances in Intelligent Systems and Computing, pages 707–728, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-38789-5_77.

[JMR+01] J. Jozefowska, M. Mika, R. Rozycki, G. Waligora, and J. Weglarz. Simulated Annealing for Multi-Mode Resource-Constrained Project Scheduling. *Annals of Operations Research*, 102(1):137–155, February 2001. doi:10.1023/A:1010954031930.

[KC13] M. Khemakhem and H. Chtourou. Efficient robustness measures for the resource-constrained project scheduling problem. *International Journal of Industrial and Systems Engineering*, 14(2):245, 2013. URL: http://www.inderscience.com/link.php?id=53738, doi:10.1504/IJISE.2013.053738.

[KD97] R. Kolisch and A. Drexl. Local search for nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, 29(11):987–999, November 1997. doi:10.1023/A:1018552303415.

[Kel14] C. Kellenbrink. *Ressourcenbeschränkte Projektplanung für flexible Projekte*. Produktion und Logistik. Springer-Gabler, Wiesbaden, 2014.

[KH98] R. Kolisch and S. Hartmann. Heuristic Algorithms for Solving the Resource-Constrained Project Scheduling Problem : Classification and Computational Analysis. Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 469, Universität Kiel, Institut für Betriebswirtschaftslehre, 1998. URL: https://www.semanticscholar.org/paper/Heuristic-Algorithms-for-Solving-the-Project-%3A-and-Kolisch-Hartmann/73fcd16cf279ee407da01a7471d4f6b61f610bae.

[KKA16] A. M. Kashtiban, S. Khanmohammadi, and K. Asghari. Solving multimodal optimization problems based on efficient partitioning of genotypic search space. *Turkish Journal of Electrical Engineering and Computer Sciences*, 2016. doi:10.3906/elk-1307-184.

[Kra17] O. Kramer. Genetic Algorithms. In *Genetic Algorithm Essentials*, volume 679. Springer International Publishing, Cham, 2017. URL: http://link.springer.com/10.1007/978-3-319-52156-5_2, doi:10.1007/978-3-319-52156-5_2.

[KS97] R. Kolisch and A. Sprecher. PSPLIB - A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program. *European Journal of Operational Research*, 96(1):205–216, January 1997. URL: https://www.sciencedirect.com/science/article/pii/S0377221796001701, doi:10.1016/S0377-2217(96)00170-1.

[LG13] Y. Li and L. Gibson. Solving MRCPSP by a Hybrid Genetic Algorithm. *Applied Mechanics and Materials*, 411-414:2369–2372, 2013. URL: https://www.scientific.net/AMM.411-414.2369, doi:10.4028/www.scientific.net/AMM.411-414.2369.

[LTB06] A. Lova, M. Tormos, and F. Barber. Multi-mode resource constrained project scheduling: scheduling schemes, priority rules and mode selection rules. *Inteligencia Artif.*, 2006. doi:10.4114/IA.V10I30.947.

[MEPKQJ04] P. Mills, T. Edward P. K., Z. Qingfu, and F. John. A survey of AI-based meta-heuristics for dealing with local optima in local search. *Technical Report Series*, CSM-416, September 2004.

[Nel07] R. Nelson. IT Project Management: Infamous Failures, Classic Mistakes, and Best Practices. *MIS Q. Executive*, 2007.

[RSMA15] J. Rezaeian, F. Soleimani, S. Mohaselafshary, and A. Arab. Using a meta-heuristic algorithm for solving the multi-mode resource-constrained project scheduling problem. *International Journal of Operational Research*, 24(1):1, 2015. URL: http://www.inderscience.com/link.php?id=70859, doi:10.1504/IJOR.2015.070859.

[SAA15] M. Sebt, M. Afshar, and Y. Alipouri. An Efficient Genetic Agorithm for Solving the Multi-Mode Resource-Constrained Project Scheduling Problem Based on Random Key Representation. *International Journal of Supply and Operations Management*, 2(3):905–924, November 2015. URL: http://www.ijsom.com/article_2545.html, doi:10.22034/2015.3.06.

[Sia16] P. Siarry, editor. *Metaheuristics*. Springer International Publishing : Imprint: Springer, Cham, 1st ed. 2016 edition, 2016.

[SR97] A. Schirmer and S. Riesenberg. Parameterized heuristics for project scheduling: Biased random sampling methods. Working Paper 456, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, 1997. URL: https://www.econstor.eu/handle/10419/177316.

[TBNA12] E. Talbi, M. Basseur, A. J. Nebro, and E. Alba. Multi-objective optimization using metaheuristics: non-standard algorithms. *International Transactions in Operational Research*, 19(1-2):283–305, 2012. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1475-3995.2011.00808.x, doi:10.1111/j.1475-3995.2011.00808.x.

[TS98] P. Thomas and S. Salhi. A Tabu Search Approach for the Resource Constrained Project Scheduling Problem. *Journal of Heuristics*, 4(2):123–139, July 1998. doi:10.1023/A:1009673512884.

[WMY14] P. Wuliang, H. Min, and H. Yongping. An improved ant algorithm for Multi-mode Resource Constrained Project Scheduling Problem. *RAIRO - Operations Research*, 48(4):595–614, October 2014. URL: https://www.rairo-ro.org/articles/ro/abs/2014/04/ro140025/ro140025.html, doi:10.1051/ro/2014025.

[Zan19] S. Zang. Executive Summary: Die wichtigsten Programmiersprachen, Frameworks, April 2019. URL: https://bytesforbusiness.com/executive-summary-die-wichtigsten-programmiersprachen-frameworks/.

Anhang A

Komponenten des beigefügten Materials

Neben der vorliegenden textuellen Ausarbeitung befindet sich auf dem beigefügten Medium im Rahmen der Masterarbeit entstandenes Material. Es handelt sich hierbei um ein lokales Git-Repository, welches aus folgenden Komponenten besteht:

MRCPSP-Framework Der Rootordner entspricht das Jetbrains IntelliJ IDEA Projekt, welches über die Entwicklungsumgebung importierbar ist. Im Ordner befindet sich die pom.xml, was die Maven Konfigurations-Datei darstellt, welche Metainformationen, Abhängigkeiten und Kompiliereinstellungen des MRCPSP-Framework beinhaltet. Zudem liegt ein generierter Maven-Wrapper (mvwn und mvwn.cmd) bei. Der Java-Sourcecode und die verwendeten Benchmark Instanzen sind im Ordner src vorzufinden.

Startskript der Evaluation Im Ordner build befindet sich das ausführbare Bash-Skript start_evaluation.sh, welches alle Experimente gemäß Abschnitt 5.1 nacheinander startet. Das gebaute Artefakt mrcpsp_framework−0.0.1−SNAPSHOT.jar liegt zudem im selben Ordner bei. Das Artefakt benötigt das Verzeichnis resources, welches die Benchmark-Instanzen beinhaltet und das Verzeichnis results in welchem die erhobenen CSV-Dateien gespeichert werden. Das Bash-Script kann über das Terminal mit bash ./start_evalution.sh oder im Hintergrund (z. B. relevant für eine entfernte Serverinstanz) mit screen −S masterthesis −m −d bash ./start_evalution.sh gestartet werden.

Erhobene CSV-Daten Die erhobenen Daten befinden sich kategorisiert im Ordner evaluation/data.

Jupyter Notebooks Im Rahmen der Evaluation wurden für die verwendeten Ergebnisse, Tabellen und Abbildungen Python Skripte erstellt. Die für die Experimente jeweils zuordenbare Skripte befinden sich im Ordner evaluation. Der

erste Codeblock nach den Import-Direktiven definiert die Pfad-Variable der zu analysierenden CSV-Dateien. Bei der Reproduktion gilt es diese anzupassen.

LaTeX-Code und PDF-Datei der Thesis Diese schriftliche Ausarbeitung wurde in LaTeX geschrieben. Der Sourecode und die PDF-Datei der Thesis liegen im Ordner thesis vor.

Anhang B

UML-Klassendiagramme

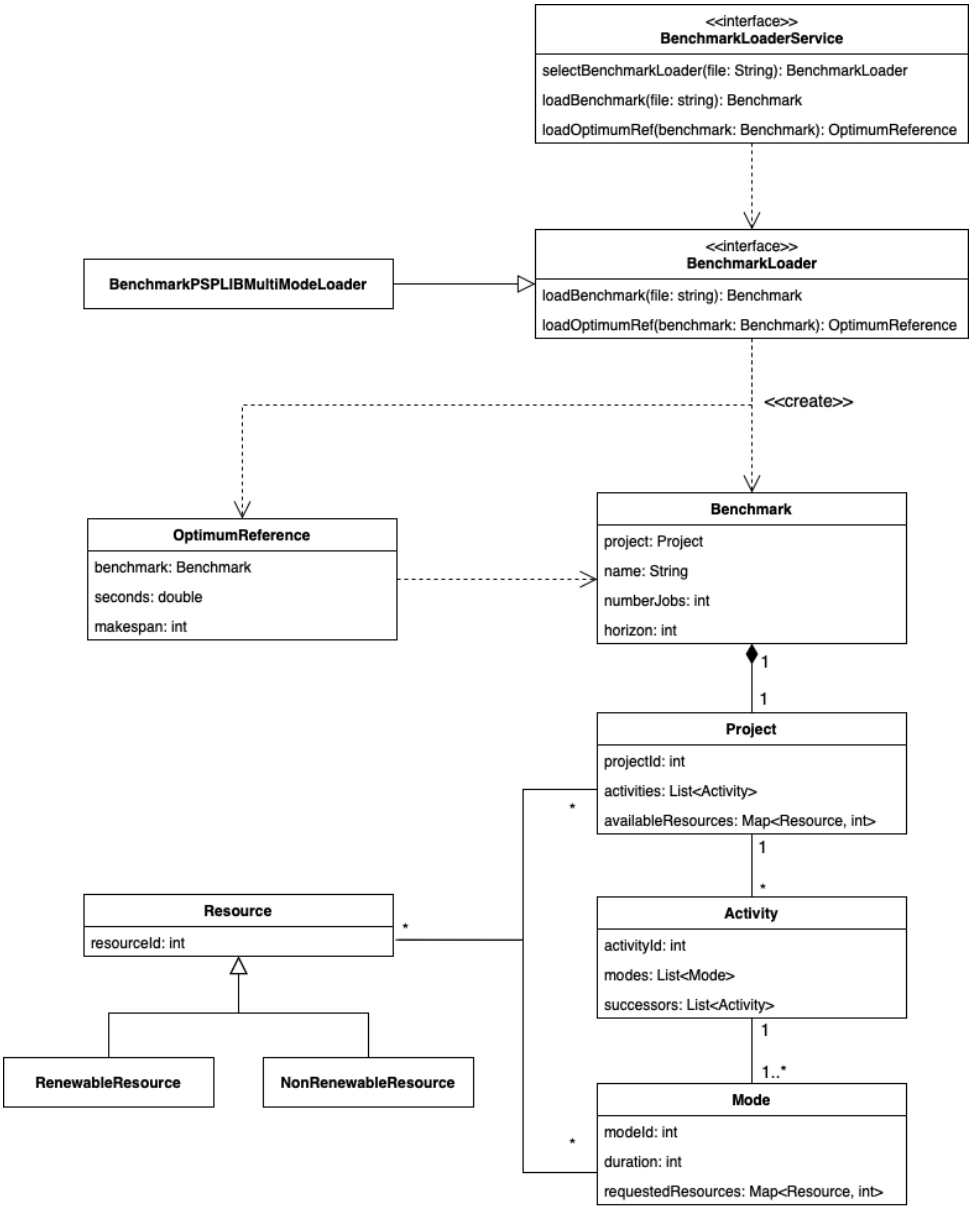


Abbildung B.1.: UML-Klassendiagramm für den Benchmark Loader

Quelle: Eigene Darstellung

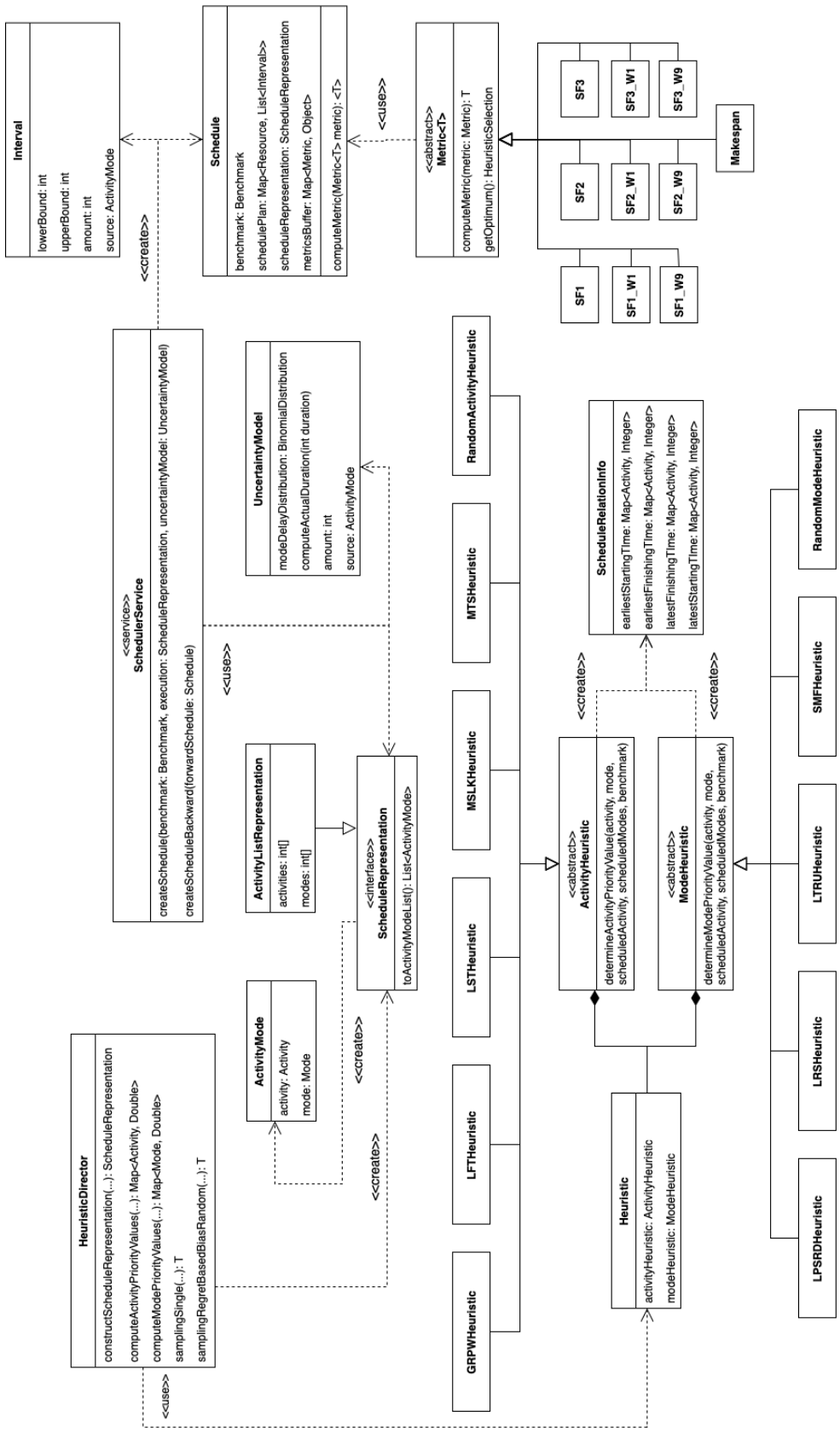


Abbildung B.2.: UML-Klassendiagramm für den Scheduler und Heuristiken

Quelle: Eigene Darstellung

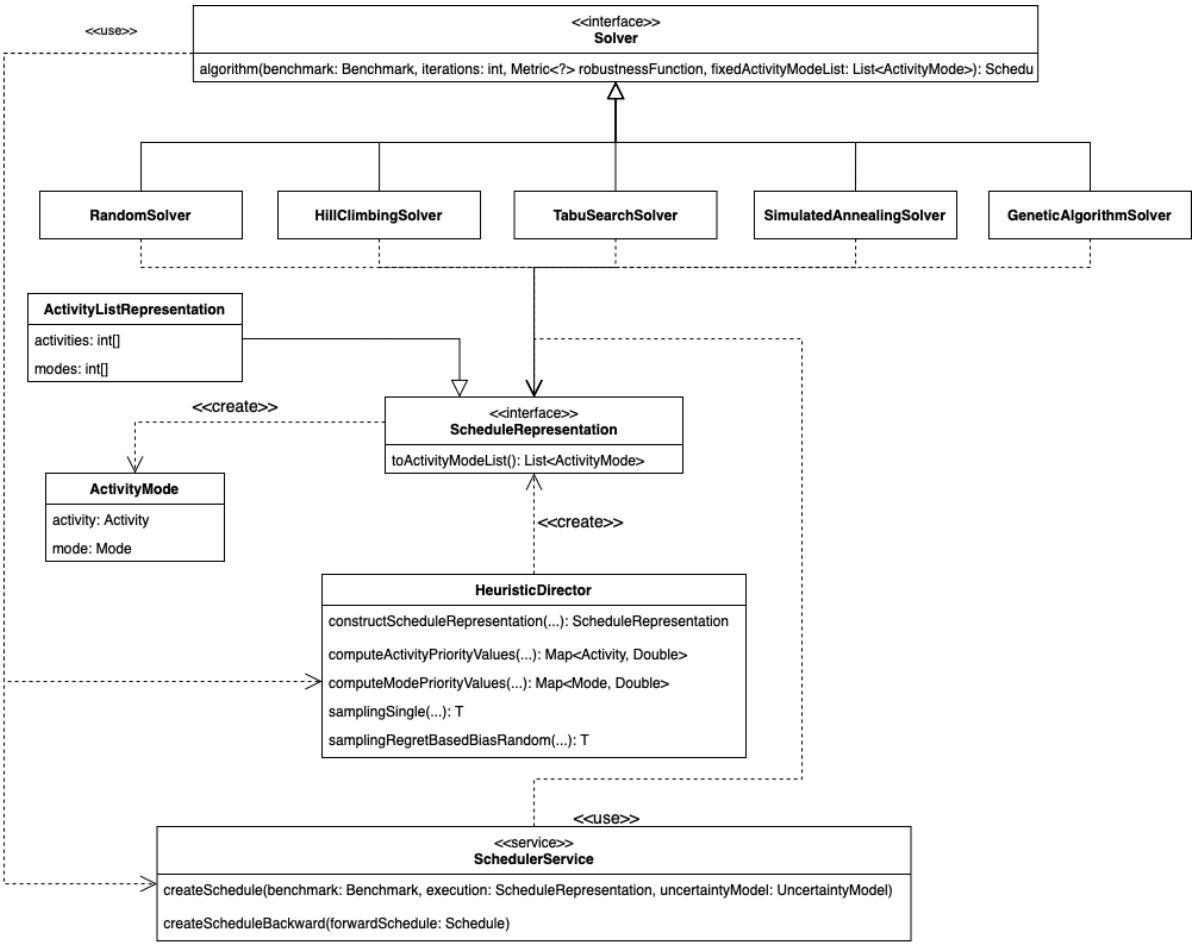


Abbildung B.3.: UML-Klassendiagramm für die Solver

Quelle: Eigene Darstellung

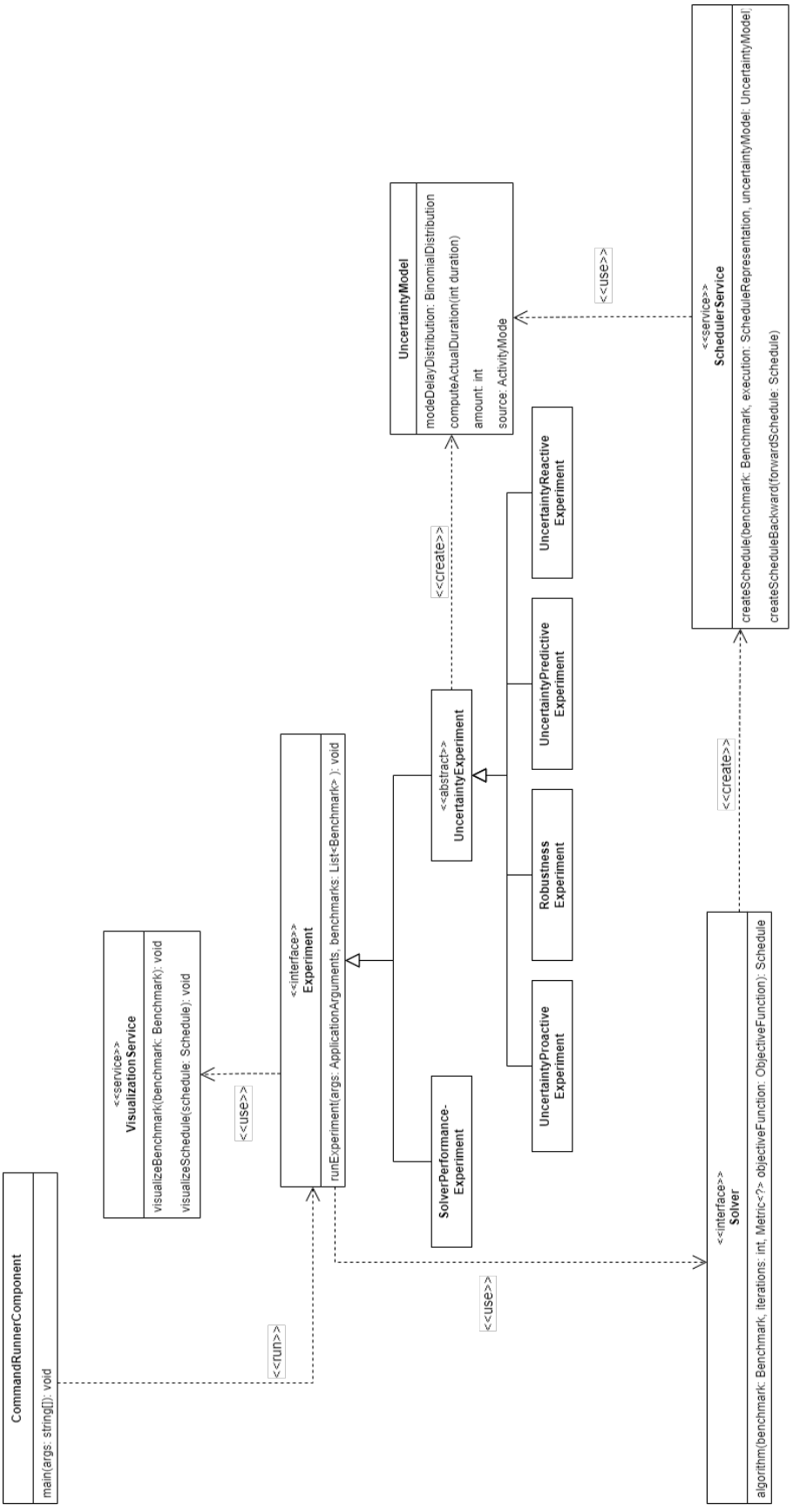


Abbildung B.4.: UML-Klassendiagramm für die Experimente

Quelle: Eigene Darstellung

Anhang C

Quellcode

Listing C.1: Erzeugen von Zeitplänen anhand von Aktivitäts- und Moduslisten

```
1  for (ActivityMode activityMode : activityModeList) {
2    Mode currentMode = activityMode.getMode();
3    // Create initial interval
4    int potentialLowerBound = earliestStartTime.getOrDefault(activityMode.getActivity(), 0);
5
6    // Construct the solution according the execution
7    boolean solutionFound = false;
8    while (!solutionFound) {
9      solutionFound = true;
10     for (Map.Entry<Resource, Integer> entry : currentMode.getRequestedResources().entrySet()) {
11       Resource currentModeResource = entry.getKey();
12       Integer currentModeAmount = entry.getValue();
13
14       int resourceAvailableGeneral = benchmark.getProject().getAvailableResources()
15                                      .get(currentModeResource);
16       int resourceAvailableOnInterval;
17
18       if (currentModeResource instanceof RenewableResource) {
19         int potentialUpperBound = potentialLowerBound + modeDurations.get(activityMode) - 1;
20
21         // potential new interval that needs to be checked
22         Interval potentialInterval = new Interval(potentialLowerBound,
23                   potentialUpperBound,
24                   currentModeAmount,
25                   activityMode);
26
27         resourceAvailableOnInterval = this.computeAvailableResourcesOnInterval(schedulePlan,
28             currentModeResource,
29             potentialInterval,
30             resourceAvailableGeneral);
31       } else {
32         resourceAvailableOnInterval = nonRenewableResourcesLeft.get(currentModeResource);
33       }
34
35       if (currentModeAmount > resourceAvailableGeneral) {
36         throw new RenewableResourceNotEnoughException();
37       } else if (currentModeAmount > resourceAvailableOnInterval &&
38                 currentModeResource instanceof NonRenewableResource) {
39             throw new NoNonRenewableResourcesLeftException(activityMode.getActivity());
40       } else if (resourceAvailableOnInterval - currentModeAmount < 0) {
41         solutionFound = false;
42         potentialLowerBound++;
```

```
43            break;
44          }
45        }
46      }
47
48      this.addInterval(activityMode,
49        modeDurations.get(activityMode),
50        schedulePlan,
51        nonRenewableResourcesLeft,
52        earliestStartTime,
53        potentialLowerBound);
54  }
```

Listing C.2: Algorithmus zur Berechnung von vorhandenen Ressourcen einer Ressourcenart für potentielles Intervall

```
1   private int computeAvailableResourcesOnInterval(Map<Resource, List<Interval>> scheduledPlan,
2                                                   Resource currentModeResource,
3                                                   Interval potentialInterval,
4                                                   int resourceAvailableGeneral) {
5     int resourceAvailableOnInterval = resourceAvailableGeneral;
6     List<Interval> conflictingIntervals = new ArrayList<>();
7
8     // determine the actual resource availability on the given interval
9     for (Interval intervalToCheck : scheduledPlan.get(currentModeResource)) {
10      if (potentialInterval.conflictInterval(intervalToCheck)) {
11        conflictingIntervals.add(intervalToCheck);
12      }
13    }
14
15    // Calculate highest available resource at every timepoint
16    for (int i = potentialInterval.getLowerBound(); i <= potentialInterval.getUpperBound(); i++) {
17      int finalI = i;
18      List<Interval> conflictingIntervalAtTimeslot = conflictingIntervals.stream()
19              .filter(interval -> interval.valueInInterval(finalI)).collect(Collectors.toList());
20      int availableAtTimeslot = resourceAvailableGeneral - conflictingIntervalAtTimeslot.stream()
21              .map(Interval::getAmount).reduce((Integer::sum)).orElse(0);
22      resourceAvailableOnInterval = Math.min(resourceAvailableOnInterval, availableAtTimeslot);
23    }
24
25    return resourceAvailableOnInterval;
26  }
```

Anhang D

Auswertungen

D.1. Weitere Auswertung der implementierten Metaheuristiken

Instance set m1		Dev. Makespan C^{Δ}_{max}		Robustness Ω^{SF1}		Feasible	Optimum
		Mean μ	Std. Dev. σ	Mean μ	Std. Dev. σ	in %	
Solver	Iteration						
RandomSolver	500	0.06	0.29	21.87	6.61	100.00	98.90
	1000	0.03	0.20	22.04	6.59	100.00	99.69
	2500	0.01	0.11	22.19	6.57	100.00	100.00
	5000	0.00	0.07	22.33	6.53	100.00	99.84
HillClimbing	500	1.83	2.96	21.11	6.34	100.00	63.27
	1000	1.85	2.98	21.16	6.38	100.00	63.42
	2500	1.85	2.99	21.09	6.39	100.00	62.64
	5000	1.85	2.97	21.17	6.39	100.00	62.32
TabuSearch	500	1.37	2.50	21.33	6.42	100.00	69.70
	1000	1.39	2.48	21.33	6.38	100.00	69.39
	2500	1.37	2.48	21.36	6.39	100.00	70.33
	5000	1.38	2.50	21.35	6.43	100.00	69.70
SimulatedAnnealing	500	1.31	2.43	21.53	6.30	100.00	76.77
	1000	1.14	2.21	21.66	6.30	100.00	80.38
	2500	0.88	1.91	21.90	6.24	100.00	84.62
	5000	0.70	1.71	21.96	6.28	100.00	88.70
GeneticAlgorithm	500	0.16	0.67	22.10	6.43	100.00	97.80
	1000	0.14	0.63	22.09	6.44	100.00	98.59
	2500	0.13	0.61	22.16	6.38	100.00	98.43
	5000	0.14	0.65	22.23	6.41	100.00	98.27

Tabelle D.1.: Vergleich der Lösungsverfahren für das Instanzset m1

Quelle: Eigene Darstellung

Instance set m2		Dev. Makespan C^{Δ}_{max}		Robustness Ω^{SF1}		Feasible	Optimum
		Mean μ	Std. Dev. σ	Mean μ	Std. Dev. σ	in %	
Solver	Iteration						
RandomSolver	500	2.76	3.43	17.57	5.87	99.79	41.04
	1000	2.26	3.11	18.23	5.68	100.00	47.50
	2500	1.62	2.59	18.26	5.97	100.00	59.38
	5000	1.30	2.17	18.45	6.27	100.00	63.96
HillClimbing	500	2.29	2.68	20.13	6.09	100.00	56.67
	1000	2.29	2.79	19.97	6.25	100.00	57.92
	2500	2.29	2.77	20.03	6.20	100.00	57.71
	5000	2.24	2.71	20.09	6.09	100.00	59.38
TabuSearch	500	1.65	2.27	20.39	6.34	100.00	70.42
	1000	1.35	2.03	20.21	6.28	100.00	78.12
	2500	1.05	1.80	20.20	6.39	100.00	86.04
	5000	0.86	1.71	20.21	6.42	100.00	89.38
SimulatedAnnealing	500	2.02	2.61	20.10	6.39	100.00	64.17
	1000	1.42	2.03	20.03	6.48	100.00	73.96
	2500	0.87	1.53	19.96	6.52	100.00	85.62
	5000	0.60	1.23	20.01	6.63	100.00	92.08
GeneticAlgorithm	500	1.36	1.74	19.52	6.50	100.00	61.25
	1000	0.64	1.09	19.62	6.57	100.00	82.71
	2500	0.40	0.84	19.87	6.48	100.00	89.17
	5000	0.33	0.75	19.76	6.52	100.00	92.50

Tabelle D.2.: Vergleich der Lösungsverfahren für das Instanzset m2

Quelle: Eigene Darstellung

Instance set n0		Dev. Makespan C^{Δ}_{max}		Robustness Ω^{SF1}		Feasible	Optimum
		Mean μ	Std. Dev. σ	Mean μ	Std. Dev. σ	in %	
Solver	Iteration						
RandomSolver	500	4.64	3.38	11.42	4.43	100.00	20.68
	1000	3.95	3.14	11.49	4.40	100.00	26.44
	2500	3.17	2.83	11.78	5.21	100.00	33.48
	5000	2.73	2.69	12.18	5.28	100.00	42.22
HillClimbing	500	1.30	2.16	15.94	6.66	100.00	78.46
	1000	1.17	2.01	16.25	6.91	100.00	78.46
	2500	1.09	1.94	16.02	6.81	100.00	80.60
	5000	1.10	1.96	16.13	6.81	100.00	80.60
TabuSearch	500	0.82	1.65	15.96	7.22	100.00	83.58
	1000	0.56	1.30	16.29	7.26	100.00	87.63
	2500	0.30	0.85	16.44	7.34	100.00	92.96
	5000	0.19	0.62	16.53	7.37	100.00	95.74
SimulatedAnnealing	500	1.60	2.25	15.45	6.80	100.00	72.71
	1000	0.83	1.50	16.01	7.21	100.00	80.38
	2500	0.40	0.90	16.29	7.54	100.00	89.77
	5000	0.23	0.59	16.34	7.61	100.00	92.32
GeneticAlgorithm	500	0.60	1.18	15.84	6.75	100.00	80.38
	1000	0.36	0.83	16.17	7.13	100.00	87.21
	2500	0.23	0.59	16.41	7.43	100.00	93.18
	5000	0.21	0.58	16.47	7.46	100.00	93.18

Tabelle D.3.: Vergleich der Lösungsverfahren für das Instanzset n0

Quelle: Eigene Darstellung

Instance set j20		Dev. Makespan C^{Δ}_{max}		Robustness Ω^{SF1}		Feasible	Optimum
		Mean μ	Std. Dev. σ	Mean μ	Std. Dev. σ	in %	
Solver	Iteration						
RandomSolver	500	7.18	9.29	16.19	4.52	99.28	2.35
	1000	7.05	7.17	18.62	5.42	99.28	3.25
	2500	6.25	5.70	17.94	5.00	99.28	6.14
	5000	5.56	5.39	18.73	4.86	99.28	7.40
HillClimbing	500	3.32	3.57	22.05	6.19	100.00	43.50
	1000	3.14	3.43	22.18	6.17	100.00	42.06
	2500	3.10	3.37	21.77	6.21	100.00	45.49
	5000	3.08	3.41	21.96	6.23	100.00	45.49
TabuSearch	500	2.75	3.13	22.12	6.57	100.00	48.92
	1000	2.26	2.81	22.20	6.80	100.00	55.23
	2500	1.76	2.46	22.20	6.86	100.00	62.82
	5000	1.40	2.08	22.12	7.12	100.00	70.22
SimulatedAnnealing	500	3.67	3.60	21.63	6.18	100.00	39.71
	1000	2.65	2.97	22.00	6.75	100.00	48.92
	2500	1.71	2.17	22.15	7.21	100.00	59.57
	5000	1.19	1.70	21.70	7.49	100.00	69.86
GeneticAlgorithm	500	3.43	3.69	21.25	6.05	100.00	38.81
	1000	1.92	2.21	21.75	6.78	100.00	49.10
	2500	0.97	1.30	21.53	7.36	100.00	66.25
	5000	0.72	1.09	21.38	7.48	100.00	75.99

Tabelle D.4.: Vergleich der Lösungsverfahren für das Instanzset j20

Quelle: Eigene Darstellung

D.2. Weitere Auswertung der proaktiven Verfahren

Instance set m1		Mean μ				Std. Dev σ			
	Uncertainty p:	0%	5%	10%	20%	0%	5%	10%	20%
Solver	Iteration								
RandomSolver	500	0.05	0.93	1.75	3.26	0.24	1.33	1.76	2.20
	1000	0.05	0.91	1.71	3.20	0.25	1.31	1.69	2.15
	2500	0.03	0.92	1.73	3.26	0.16	1.42	1.80	2.25
	5000	0.02	0.92	1.72	3.22	0.14	1.35	1.75	2.21
HillClimbing	500	1.78	2.56	3.33	4.78	2.46	2.73	2.96	3.28
	1000	1.85	2.65	3.39	4.84	2.58	2.82	3.05	3.40
	2500	1.80	2.60	3.36	4.83	2.53	2.80	3.01	3.38
	5000	1.84	2.62	3.38	4.85	2.56	2.81	3.00	3.37
TabuSearch	500	1.40	2.21	2.99	4.47	2.12	2.43	2.69	3.06
	1000	1.34	2.17	2.94	4.42	2.07	2.42	2.67	3.04
	2500	1.33	2.16	2.93	4.44	2.04	2.40	2.64	3.06
	5000	1.37	2.20	2.97	4.46	2.16	2.51	2.74	3.13
SimulatedAnnealing	500	1.44	2.27	3.04	4.48	2.34	2.65	2.84	3.17
	1000	1.24	2.03	2.78	4.24	2.06	2.35	2.56	2.89
	2500	0.92	1.73	2.48	3.93	1.84	2.17	2.40	2.76
	5000	0.72	1.55	2.32	3.78	1.65	2.01	2.27	2.63
GeneticAlgorithm	500	0.14	1.05	1.84	3.34	0.59	1.56	1.94	2.43
	1000	0.08	1.01	1.85	3.36	0.36	1.61	2.08	2.58
	2500	0.09	1.02	1.84	3.38	0.43	1.70	2.13	2.64
	5000	0.09	0.97	1.79	3.31	0.41	1.45	1.87	2.38

Tabelle D.5.: Vergleich der proaktiven Lösungsverfahren auf $m = 50$ unterschiedliche Unsicherheitsszenarien für das Instanzset m1.

Quelle: Eigene Darstellung

Instance set m2		Mean μ				Std. Dev σ			
	Uncertainty p:	0%	5%	10%	20%	0%	5%	10%	20%
Solver	Iteration								
RandomSolver	500	2.96	3.83	4.61	6.11	3.17	3.55	3.80	4.18
	1000	2.32	3.21	4.05	5.58	2.67	3.10	3.39	3.78
	2500	1.65	2.57	3.40	4.92	2.29	2.79	3.04	3.44
	5000	1.29	2.21	3.03	4.57	1.86	2.42	2.77	3.20
HillClimbing	500	2.93	3.78	4.58	6.00	3.45	3.71	3.91	4.15
	1000	3.04	3.90	4.67	6.10	3.50	3.78	3.95	4.20
	2500	2.90	3.77	4.56	5.99	3.54	3.83	4.05	4.31
	5000	2.87	3.78	4.58	6.04	3.40	3.80	4.02	4.24
TabuSearch	500	1.37	2.25	3.02	4.49	1.90	2.31	2.57	2.88
	1000	1.13	1.99	2.79	4.26	1.90	2.22	2.42	2.75
	2500	0.77	1.65	2.46	3.93	1.45	1.88	2.17	2.51

	5000	0.67	1.55	2.35	3.83	1.38	1.78	2.04	2.35
SimulatedAnnealing	500	1.89	2.74	3.50	4.99	2.31	2.60	2.80	3.11
	1000	1.19	2.08	2.90	4.35	1.75	2.17	2.45	2.73
	2500	0.74	1.60	2.38	3.83	1.25	1.70	1.93	2.30
	5000	0.50	1.41	2.24	3.67	1.04	1.73	2.18	2.48
GeneticAlgorithm	500	1.40	2.22	2.99	4.46	1.89	2.28	2.54	2.95
	1000	0.63	1.46	2.24	3.70	1.11	1.67	2.02	2.42
	2500	0.36	1.19	1.98	3.45	0.74	1.40	1.76	2.16
	5000	0.26	1.10	1.90	3.34	0.64	1.33	1.74	2.14

Tabelle D.6.: Vergleich der proaktiven Lösungsverfahren auf $m = 50$ unterschiedliche Unsicherheitsszenarien für das Instanzset m2.

Quelle: Eigene Darstellung

Instance set n0		Mean μ				Std. Dev σ			
	Uncertainty p:	0%	5%	10%	20%	0%	5%	10%	20%
Solver	Iteration								
RandomSolver	500	4.48	5.30	6.06	7.51	3.24	3.58	3.85	4.29
	1000	3.82	4.63	5.40	6.78	3.07	3.48	3.80	4.17
	2500	3.09	3.95	4.71	6.15	2.81	3.29	3.55	4.08
	5000	2.62	3.46	4.22	5.67	2.62	2.97	3.26	3.73
HillClimbing	500	3.19	4.15	4.97	6.44	3.69	4.13	4.44	4.85
	1000	3.21	4.21	5.07	6.52	3.90	4.44	4.81	5.20
	2500	3.31	4.28	5.11	6.69	3.84	4.38	4.63	5.14
	5000	3.11	4.10	4.94	6.47	3.69	4.20	4.55	5.05
TabuSearch	500	1.01	1.94	2.71	4.22	1.85	2.48	2.80	3.38
	1000	0.57	1.47	2.26	3.70	1.32	1.95	2.31	2.80
	2500	0.24	1.17	1.97	3.44	0.73	1.74	2.13	2.61
	5000	0.12	1.07	1.88	3.33	0.45	1.54	1.93	2.38
SimulatedAnnealing	500	1.97	2.88	3.67	5.13	2.60	3.08	3.40	3.80
	1000	1.05	1.97	2.77	4.21	1.81	2.33	2.67	3.05
	2500	0.42	1.32	2.11	3.57	1.12	1.72	2.04	2.56
	5000	0.24	1.17	1.98	3.48	0.70	1.47	1.85	2.31
GeneticAlgorithm	500	0.57	1.38	2.14	3.54	1.19	1.72	2.07	2.54
	1000	0.26	1.09	1.83	3.28	0.67	1.39	1.74	2.25
	2500	0.16	0.99	1.75	3.18	0.46	1.25	1.66	2.17
	5000	0.13	0.97	1.70	3.15	0.44	1.24	1.58	2.10

Tabelle D.7.: Vergleich der proaktiven Lösungsverfahren auf $m = 50$ unterschiedliche Unsicherheitsszenarien für das Instanzset n0.

Quelle: Eigene Darstellung

D.3. Weitere Auswertung der prädiktiven Verfahren

D.3.1. Vergleich der unterschiedlichen Robustheitsmessungen

Instance set n1		Mean μ				Std. Dev σ			
RandomSolver Iteration	Uncertainty p: $\Omega(x)$	0%	5%	10%	20%	0%	5%	10%	20%
500	No-RM	0.00	0.82	1.62	3.09	0.00	1.16	1.62	2.07
	SF1	0.00	0.85	1.65	3.12	0.00	1.26	1.72	2.23
	SF1_W1	0.00	0.83	1.61	3.06	0.00	1.24	1.65	2.11
	SF1_W9	0.00	0.83	1.61	3.11	0.00	1.18	1.58	2.05
	SF2	0.00	0.83	1.59	3.03	0.00	1.22	1.59	2.06
	SF2_W1	0.00	0.82	1.56	3.05	0.00	1.27	1.65	2.12
	SF2_W9	0.00	0.84	1.62	3.08	0.00	1.32	1.71	2.17
	SF3	0.00	0.85	1.63	3.15	0.00	1.31	1.71	2.22
	SF3_W1	0.00	0.83	1.62	3.06	0.00	1.22	1.65	2.06
	SF3_W9	0.00	0.80	1.58	3.04	0.00	1.07	1.46	1.90
1000	No-RM	0.00	0.88	1.69	3.22	0.00	1.32	1.73	2.22
	SF1	0.00	0.81	1.60	3.09	0.00	1.18	1.63	2.13
	SF1_W1	0.00	0.89	1.69	3.23	0.00	1.39	1.75	2.26
	SF1_W9	0.00	0.85	1.65	3.13	0.00	1.29	1.70	2.18
	SF2	0.00	0.81	1.58	3.04	0.00	1.18	1.56	2.00
	SF2_W1	0.00	0.79	1.54	3.02	0.00	1.06	1.45	1.95
	SF2_W9	0.00	0.83	1.62	3.07	0.00	1.18	1.59	2.03
	SF3	0.00	0.82	1.56	3.04	0.00	1.17	1.48	2.02
	SF3_W1	0.00	0.81	1.60	3.08	0.00	1.12	1.50	1.99
	SF3_W9	0.00	0.81	1.61	3.08	0.00	1.16	1.60	2.06
2500	No-RM	0.00	0.86	1.67	3.19	0.00	1.24	1.64	2.18
	SF1	0.00	0.82	1.61	3.07	0.00	1.19	1.57	1.98
	SF1_W1	0.00	0.87	1.66	3.16	0.00	1.22	1.65	2.06
	SF1_W9	0.00	0.86	1.65	3.14	0.00	1.28	1.65	2.12
	SF2	0.00	0.82	1.58	3.05	0.00	1.24	1.61	2.07
	SF2_W1	0.00	0.81	1.59	3.09	0.00	1.14	1.55	2.02
	SF2_W9	0.00	0.82	1.60	3.06	0.00	1.17	1.57	2.04
	SF3	0.00	0.83	1.62	3.07	0.00	1.16	1.53	1.96
	SF3_W1	0.00	0.83	1.63	3.11	0.00	1.21	1.70	2.14
	SF3_W9	0.00	0.87	1.67	3.16	0.00	1.19	1.58	2.02
5000	No-RM	0.00	0.89	1.70	3.18	0.00	1.27	1.63	2.09
	SF1	0.00	0.86	1.67	3.17	0.00	1.21	1.64	2.04
	SF1_W1	0.00	0.86	1.65	3.15	0.00	1.25	1.66	2.11
	SF1_W9	0.00	0.84	1.64	3.15	0.00	1.17	1.57	2.02
	SF2	0.00	0.83	1.62	3.11	0.00	1.23	1.60	2.12
	SF2_W1	0.00	0.83	1.61	3.09	0.00	1.14	1.53	1.98
	SF2_W9	0.00	0.87	1.68	3.20	0.00	1.24	1.63	2.14
	SF3	0.00	0.84	1.63	3.12	0.00	1.21	1.60	2.04
	SF3_W1	0.00	0.84	1.64	3.15	0.00	1.17	1.60	2.04
	SF3_W9	0.00	0.91	1.72	3.23	0.00	1.41	1.82	2.29

Tabelle D.8.: Verspätungswerte der Robustheitfunktionen $\Omega(x)$ angewendet auf den RS für das Instanzset n1 mit $n = 50$ Unsicherheitsszenarien.

Quelle: Eigene Darstellung

Instance set n1		Mean μ				Std. Dev σ			
HillClimbing Iteration	Uncertainty p: $\Omega(x)$	0%	5%	10%	20%	0%	5%	10%	20%
500	No-RM	0.00	0.97	1.81	3.33	0.00	1.27	1.62	2.03
	SF1	0.00	0.92	1.72	3.26	0.00	1.41	1.78	2.28
	SF1_W1	0.00	0.87	1.68	3.17	0.00	1.21	1.63	2.11
	SF1_W9	0.00	0.93	1.75	3.28	0.00	1.35	1.75	2.25
	SF2	0.00	0.87	1.64	3.16	0.00	1.32	1.76	2.25
	SF2_W1	0.00	0.85	1.61	3.13	0.00	1.26	1.66	2.17
	SF2_W9	0.00	0.82	1.59	3.09	0.00	1.20	1.59	2.11
	SF3	0.00	0.91	1.75	3.30	0.00	1.43	1.86	2.39
	SF3_W1	0.00	0.87	1.66	3.19	0.00	1.23	1.62	2.12
	SF3_W9	0.00	0.89	1.68	3.21	0.00	1.28	1.65	2.20
1000	No-RM	0.00	0.96	1.80	3.34	0.00	1.16	1.52	2.02
	SF1	0.00	0.94	1.78	3.31	0.00	1.39	1.79	2.29
	SF1_W1	0.00	0.88	1.68	3.19	0.00	1.16	1.57	2.05
	SF1_W9	0.00	0.90	1.71	3.22	0.00	1.32	1.72	2.23
	SF2	0.00	0.82	1.61	3.07	0.00	1.13	1.54	2.01
	SF2_W1	0.00	0.83	1.59	3.13	0.00	1.16	1.52	2.10
	SF2_W9	0.00	0.87	1.68	3.18	0.00	1.28	1.71	2.17
	SF3	0.00	0.87	1.69	3.19	0.00	1.22	1.65	2.09
	SF3_W1	0.00	0.90	1.73	3.27	0.00	1.33	1.70	2.21
	SF3_W9	0.00	0.89	1.72	3.24	0.00	1.30	1.70	2.18
2500	No-RM	0.00	0.96	1.80	3.35	0.00	1.22	1.63	2.16
	SF1	0.00	0.95	1.75	3.29	0.00	1.43	1.79	2.28
	SF1_W1	0.00	0.88	1.70	3.20	0.00	1.25	1.66	2.14
	SF1_W9	0.00	0.93	1.75	3.25	0.00	1.33	1.71	2.20
	SF2	0.00	0.87	1.66	3.17	0.00	1.25	1.68	2.31
	SF2_W1	0.00	0.85	1.62	3.13	0.00	1.25	1.61	2.13
	SF2_W9	0.00	0.85	1.64	3.15	0.00	1.19	1.56	2.07
	SF3	0.00	0.91	1.72	3.25	0.00	1.33	1.72	2.21
	SF3_W1	0.00	0.92	1.72	3.24	0.00	1.43	1.80	2.22
	SF3_W9	0.00	0.87	1.69	3.19	0.00	1.28	1.64	2.11
5000	No-RM	0.00	0.99	1.87	3.40	0.00	1.35	1.72	2.14
	SF1	0.00	0.88	1.68	3.17	0.00	1.21	1.59	2.08
	SF1_W1	0.00	0.92	1.75	3.24	0.00	1.38	1.78	2.26
	SF1_W9	0.00	0.88	1.70	3.19	0.00	1.20	1.59	2.08
	SF2	0.00	0.84	1.64	3.11	0.00	1.21	1.65	2.09
	SF2_W1	0.00	0.83	1.61	3.09	0.00	1.24	1.59	2.05
	SF2_W9	0.00	0.87	1.67	3.16	0.00	1.28	1.64	2.15
	SF3	0.00	0.90	1.70	3.24	0.00	1.30	1.68	2.20
	SF3_W1	0.00	0.86	1.68	3.15	0.00	1.15	1.52	1.93
	SF3_W9	0.00	0.89	1.71	3.22	0.00	1.37	1.73	2.24

Tabelle D.9.: Verspätungswerte der Robustheitfunktionen $\Omega(x)$ angewendet auf HC für das Instanzset n1 mit $n = 50$ Unsicherheitsszenarien.

Quelle: Eigene Darstellung

Instance set n1		Mean μ				Std. Dev σ			
SimulatedAnnealing Iteration	Uncertainty p: $\Omega(x)$	0%	5%	10%	20%	0%	5%	10%	20%
500	No-RM	0.00	0.93	1.74	3.28	0.00	1.25	1.63	2.12
	SF1	0.00	0.86	1.65	3.09	0.00	1.15	1.49	1.95
	SF1_W1	0.00	0.86	1.66	3.14	0.00	1.21	1.56	2.04
	SF1_W9	0.00	0.90	1.69	3.18	0.00	1.26	1.64	2.15
	SF2	0.00	0.82	1.61	3.09	0.00	1.16	1.58	2.06
	SF2_W1	0.00	0.84	1.64	3.10	0.00	1.23	1.60	2.08
	SF2_W9	0.00	0.86	1.66	3.14	0.00	1.29	1.69	2.17
	SF3	0.00	0.82	1.60	3.07	0.00	1.09	1.44	1.90
	SF3_W1	0.00	0.85	1.66	3.14	0.00	1.17	1.56	2.07
	SF3_W9	0.00	0.83	1.64	3.13	0.00	1.21	1.61	2.11
1000	No-RM	0.00	0.98	1.80	3.34	0.00	1.20	1.55	1.98
	SF1	0.00	0.87	1.65	3.12	0.00	1.15	1.51	1.93
	SF1_W1	0.00	0.89	1.66	3.15	0.00	1.26	1.61	2.10
	SF1_W9	0.00	0.88	1.69	3.14	0.00	1.20	1.58	2.00
	SF2	0.00	0.85	1.62	3.10	0.00	1.21	1.58	2.06
	SF2_W1	0.00	0.85	1.64	3.14	0.00	1.16	1.54	2.05
	SF2_W9	0.00	0.91	1.72	3.25	0.00	1.40	1.77	2.27
	SF3	0.00	0.86	1.68	3.18	0.00	1.23	1.67	2.12
	SF3_W1	0.00	0.87	1.67	3.16	0.00	1.11	1.52	1.92
	SF3_W9	0.00	0.85	1.69	3.18	0.00	1.17	1.64	2.08
2500	No-RM	0.00	0.97	1.82	3.34	0.00	1.24	1.59	2.01
	SF1	0.00	0.91	1.74	3.26	0.00	1.27	1.67	2.13
	SF1_W1	0.00	0.91	1.74	3.23	0.00	1.27	1.66	2.10
	SF1_W9	0.00	0.93	1.74	3.28	0.00	1.32	1.68	2.13
	SF2	0.00	0.83	1.61	3.10	0.00	1.10	1.48	1.90
	SF2_W1	0.00	0.87	1.69	3.19	0.00	1.24	1.61	2.12
	SF2_W9	0.00	0.88	1.66	3.20	0.00	1.25	1.59	2.11
	SF3	0.00	0.87	1.70	3.18	0.00	1.22	1.61	2.03
	SF3_W1	0.00	0.89	1.70	3.21	0.00	1.22	1.61	2.07
	SF3_W9	0.00	0.89	1.70	3.21	0.00	1.27	1.70	2.15
5000	No-RM	0.00	0.97	1.81	3.33	0.00	1.15	1.53	1.93
	SF1	0.00	0.92	1.74	3.24	0.00	1.21	1.60	2.04
	SF1_W1	0.00	0.89	1.69	3.16	0.00	1.22	1.56	2.01
	SF1_W9	0.00	0.91	1.73	3.21	0.00	1.20	1.55	1.99
	SF2	0.00	0.84	1.64	3.16	0.00	1.11	1.48	1.93
	SF2_W1	0.00	0.84	1.62	3.10	0.00	1.08	1.42	1.86
	SF2_W9	0.00	0.85	1.66	3.17	0.00	1.18	1.56	2.04
	SF3	0.00	0.88	1.71	3.25	0.00	1.18	1.61	2.05
	SF3_W1	0.00	0.87	1.69	3.19	0.00	1.20	1.63	2.07
	SF3_W9	0.00	0.89	1.67	3.21	0.00	1.22	1.58	2.04

Tabelle D.10.: Verspätungswerte der Robustheitfunktionen $\Omega(x)$ angewendet auf SA für das Instanzset n1 mit $n = 50$ Unsicherheitsszenarien.

Quelle: Eigene Darstellung

Instance set n1		Mean μ				Std. Dev σ			
GeneticAlgorithm Iteration	Uncertainty p: $\Omega(x)$	0%	5%	10%	20%	0%	5%	10%	20%
500	No-RM	0.00	0.86	1.66	3.18	0.00	1.06	1.40	1.90
	SF1	0.00	0.84	1.60	3.11	0.00	1.05	1.40	1.87
	SF1_W1	0.00	0.83	1.61	3.08	0.00	1.08	1.45	1.86
	SF1_W9	0.00	0.84	1.62	3.11	0.00	1.10	1.51	1.92
	SF2	0.00	0.85	1.65	3.16	0.00	1.16	1.56	2.02
	SF2_W1	0.00	0.80	1.57	3.04	0.00	0.98	1.34	1.83
	SF2_W9	0.00	0.81	1.59	3.05	0.00	1.06	1.42	1.82
	SF3	0.00	0.80	1.56	3.03	0.00	1.00	1.33	1.77
	SF3_W1	0.00	0.81	1.58	3.06	0.00	1.01	1.41	1.82
	SF3_W9	0.00	0.82	1.60	3.09	0.00	1.05	1.42	1.87
1000	No-RM	0.00	0.92	1.75	3.24	0.00	1.18	1.59	1.98
	SF1	0.00	0.88	1.69	3.17	0.00	1.18	1.53	1.98
	SF1_W1	0.00	0.86	1.64	3.13	0.00	1.05	1.39	1.81
	SF1_W9	0.00	0.88	1.68	3.15	0.00	1.10	1.46	1.89
	SF2	0.00	0.84	1.61	3.07	0.00	1.07	1.40	1.80
	SF2_W1	0.00	0.82	1.61	3.09	0.00	1.07	1.42	1.85
	SF2_W9	0.00	0.85	1.64	3.12	0.00	1.09	1.45	1.85
	SF3	0.00	0.85	1.65	3.15	0.00	1.08	1.43	1.86
	SF3_W1	0.00	0.86	1.66	3.15	0.00	1.12	1.47	1.93
	SF3_W9	0.00	0.86	1.68	3.18	0.00	1.19	1.63	2.03
2500	No-RM	0.00	0.91	1.72	3.23	0.00	1.06	1.41	1.82
	SF1	0.00	0.90	1.71	3.19	0.00	1.17	1.49	1.91
	SF1_W1	0.00	0.89	1.70	3.20	0.00	1.15	1.49	1.96
	SF1_W9	0.00	0.89	1.70	3.20	0.00	1.11	1.45	1.90
	SF2	0.00	0.82	1.61	3.06	0.00	1.07	1.42	1.81
	SF2_W1	0.00	0.87	1.65	3.17	0.00	1.14	1.46	1.90
	SF2_W9	0.00	0.86	1.66	3.14	0.00	1.09	1.47	1.91
	SF3	0.00	0.87	1.67	3.17	0.00	1.12	1.47	1.91
	SF3_W1	0.00	0.83	1.63	3.07	0.00	1.01	1.35	1.72
	SF3_W9	0.00	0.84	1.64	3.14	0.00	1.03	1.40	1.81
5000	No-RM	0.00	0.94	1.78	3.29	0.00	1.18	1.54	1.90
	SF1	0.00	0.93	1.77	3.30	0.00	1.17	1.55	2.02
	SF1_W1	0.00	0.87	1.68	3.17	0.00	1.01	1.38	1.81
	SF1_W9	0.00	0.92	1.74	3.24	0.00	1.20	1.62	2.00
	SF2	0.00	0.87	1.64	3.16	0.00	1.08	1.40	1.84
	SF2_W1	0.00	0.85	1.64	3.14	0.00	1.08	1.44	1.88
	SF2_W9	0.00	0.86	1.66	3.14	0.00	1.18	1.54	1.89
	SF3	0.00	0.89	1.70	3.22	0.00	1.22	1.61	2.03
	SF3_W1	0.00	0.87	1.67	3.16	0.00	1.15	1.49	1.95
	SF3_W9	0.00	0.86	1.66	3.14	0.00	1.08	1.43	1.83

Tabelle D.11.: Verspätungswerte der Robustheitfunktionen $\Omega(x)$ angewendet auf den GA für das Instanzset n1 mit $n = 50$ Unsicherheitsszenarien.

Quelle: Eigene Darstellung

D.3.2. Vergleich der prädiktiven Verfahren über eine konkrete Robustheitsmessung

Instance set m1		Mean μ				Std. Dev σ			
	Uncertainty p:	0%	5%	10%	20%	0%	5%	10%	20%
Solver	Iteration								
RandomSolver	500	0.08	0.99	1.82	3.33	0.33	1.50	1.94	2.41
	1000	0.05	0.91	1.69	3.23	0.24	1.35	1.70	2.32
	2500	0.03	0.91	1.71	3.25	0.16	1.37	1.81	2.30
	5000	0.02	0.93	1.78	3.32	0.13	1.49	2.01	2.50
HillClimbing	500	1.80	2.57	3.33	4.79	2.51	2.77	2.98	3.32
	1000	1.81	2.58	3.31	4.78	2.54	2.79	2.98	3.31
	2500	1.76	2.54	3.30	4.74	2.50	2.77	2.97	3.29
	5000	1.83	2.59	3.36	4.77	2.59	2.82	3.03	3.37
TabuSearch	500	1.28	2.11	2.89	4.37	2.06	2.39	2.62	2.99
	1000	1.38	2.18	2.93	4.40	2.14	2.43	2.65	3.00
	2500	1.33	2.17	2.94	4.38	2.04	2.39	2.61	2.98
	5000	1.40	2.23	2.99	4.46	2.13	2.46	2.67	3.00
SimulatedAnnealing	500	1.33	2.12	2.87	4.33	2.10	2.41	2.61	2.95
	1000	1.16	1.95	2.70	4.18	2.08	2.37	2.57	2.93
	2500	0.79	1.59	2.33	3.80	1.68	2.05	2.28	2.66
	5000	0.65	1.48	2.24	3.71	1.57	1.96	2.20	2.58
GeneticAlgorithm	500	0.14	1.05	1.85	3.37	0.56	1.67	2.02	2.54
	1000	0.09	1.01	1.84	3.34	0.38	1.60	2.03	2.53
	2500	0.10	1.01	1.85	3.35	0.42	1.60	2.10	2.58
	5000	0.10	1.01	1.80	3.31	0.44	1.60	1.95	2.53

Tabelle D.12.: Vergleich der prädiktiven Lösungsverfahren auf $m = 50$ unterschiedliche Unsicherheitsszenarien für das Instanzset m1.

Quelle: Eigene Darstellung

Instance set m2		Mean μ				Std. Dev σ			
	Uncertainty p:	0%	5%	10%	20%	0%	5%	10%	20%
Solver	Iteration								
RandomSolver	500	2.74	3.59	4.34	5.84	2.81	3.19	3.42	3.77
	1000	2.26	3.10	3.88	5.36	2.45	2.84	3.11	3.48
	2500	1.65	2.52	3.33	4.83	2.09	2.57	2.89	3.25
	5000	1.34	2.19	2.99	4.50	1.90	2.42	2.75	3.14
HillClimbing	500	2.14	2.91	3.63	5.05	2.56	2.86	3.08	3.43
	1000	2.34	3.12	3.83	5.27	2.77	3.05	3.23	3.56
	2500	2.39	3.21	3.95	5.39	3.19	3.53	3.78	4.11
	5000	2.30	3.09	3.84	5.24	2.85	3.13	3.38	3.66
TabuSearch	500	1.24	2.08	2.83	4.27	1.83	2.26	2.50	2.82

	1000	1.02	1.81	2.59	4.02	1.57	1.98	2.24	2.58
	2500	0.72	1.58	2.39	3.83	1.35	1.92	2.27	2.59
	5000	0.56	1.47	2.27	3.76	1.28	1.96	2.26	2.64
SimulatedAnnealing	500	1.72	2.49	3.23	4.67	2.24	2.51	2.72	3.00
	1000	1.16	1.94	2.68	4.11	1.73	2.05	2.29	2.60
	2500	0.81	1.63	2.38	3.85	1.38	1.88	2.11	2.50
	5000	0.48	1.36	2.16	3.63	1.08	1.74	2.06	2.46
GeneticAlgorithm	500	1.33	2.11	2.85	4.28	1.82	2.19	2.46	2.81
	1000	0.64	1.46	2.22	3.67	1.12	1.71	2.07	2.53
	2500	0.29	1.13	1.88	3.34	0.64	1.47	1.79	2.31
	5000	0.24	1.09	1.89	3.34	0.61	1.51	1.94	2.39

Tabelle D.13.: Vergleich der prädiktiven Lösungsverfahren auf $m = 50$ unterschiedliche Unsicherheitsszenarien für das Instanzset m2.

Quelle: Eigene Darstellung

Instance set n0		Mean μ				Std. Dev σ			
	Uncertainty p:	0%	5%	10%	20%	0%	5%	10%	20%
Solver	Iteration								
RandomSolver	500	4.48	5.26	5.99	7.37	3.38	3.74	3.99	4.39
	1000	3.89	4.62	5.33	6.73	3.15	3.45	3.69	4.11
	2500	3.10	3.89	4.62	5.99	2.94	3.25	3.49	3.91
	5000	2.59	3.42	4.15	5.62	2.52	2.97	3.31	3.86
HillClimbing	500	1.34	2.15	2.89	4.37	2.31	2.91	3.29	3.90
	1000	1.47	2.28	3.01	4.48	2.52	3.10	3.41	3.95
	2500	1.25	2.01	2.75	4.20	2.33	2.82	3.11	3.71
	5000	1.12	1.90	2.66	4.07	2.09	2.54	2.87	3.34
TabuSearch	500	0.72	1.55	2.29	3.72	1.63	2.30	2.64	3.10
	1000	0.39	1.22	1.98	3.44	1.01	1.79	2.17	2.73
	2500	0.28	1.06	1.80	3.20	0.82	1.50	1.89	2.37
	5000	0.10	0.93	1.66	3.11	0.44	1.45	1.84	2.40
SimulatedAnnealing	500	1.67	2.45	3.18	4.56	2.39	2.81	3.04	3.48
	1000	0.78	1.56	2.31	3.66	1.50	2.11	2.51	2.88
	2500	0.36	1.16	1.92	3.36	0.92	1.64	2.09	2.65
	5000	0.17	0.91	1.68	3.07	0.58	1.25	1.67	2.14
GeneticAlgorithm	500	0.57	1.35	2.12	3.52	1.17	1.89	2.24	2.80
	1000	0.24	0.97	1.71	3.08	0.63	1.31	1.70	2.23
	2500	0.19	0.96	1.72	3.12	0.52	1.32	1.72	2.22
	5000	0.12	0.90	1.61	3.01	0.40	1.21	1.58	2.08

Tabelle D.14.: Vergleich der prädiktiven Lösungsverfahren auf $m = 50$ unterschiedliche Unsicherheitsszenarien für das Instanzset n0.

Quelle: Eigene Darstellung

D.4. Weitere Auswertung der reaktiven Verfahren

Instance set m1		Mean μ				Std. Dev σ			
	Uncertainty p:	0%	5%	10%	20%	0%	5%	10%	20%
Solver	Iteration								
RandomSolver	500	0.08	0.93	1.69	3.15	0.32	1.40	1.61	2.07
	1000	0.05	0.91	1.66	3.03	0.29	1.32	1.57	1.89
	2500	0.02	0.87	1.64	3.11	0.12	1.22	1.55	2.13
	5000	0.02	0.82	1.57	3.08	0.12	1.08	1.44	2.03
HillClimbing	500	1.88	2.63	3.34	4.67	2.53	2.72	2.96	3.15
	1000	1.82	2.48	3.27	4.70	2.60	2.74	2.95	3.16
	2500	2.09	2.83	3.48	4.92	2.77	2.94	3.13	3.45
	5000	2.04	2.78	3.50	4.83	2.72	2.92	3.11	3.37
TabuSearch	500	1.41	2.23	2.85	4.28	2.22	2.48	2.62	2.92
	1000	1.37	2.14	2.82	4.27	2.14	2.40	2.56	2.90
	2500	1.53	2.32	3.04	4.37	2.46	2.71	2.93	3.13
	5000	1.37	2.14	2.87	4.25	2.10	2.33	2.52	2.81
SimulatedAnnealing	500	1.55	2.35	3.04	4.48	2.36	2.58	2.72	3.04
	1000	1.28	2.03	2.80	4.09	2.11	2.34	2.55	2.79
	2500	0.97	1.78	2.51	3.88	1.85	2.20	2.40	2.72
	5000	0.69	1.43	2.17	3.61	1.55	1.83	2.07	2.39
GeneticAlgorithm	500	0.13	0.99	1.81	3.31	0.51	1.53	1.95	2.41
	1000	0.10	0.87	1.73	3.17	0.39	1.19	1.83	2.14
	2500	0.07	0.85	1.65	3.16	0.34	1.27	1.68	2.18
	5000	0.08	0.88	1.63	3.06	0.37	1.14	1.59	2.03

Tabelle D.15.: Vergleich der reaktiven Lösungsverfahren auf $m = 50$ unterschiedliche Unsicherheitsszenarien für das Instanzset m1.

Quelle: Eigene Darstellung

Instance set m2		Mean μ				Std. Dev σ			
	Uncertainty p:	0%	5%	10%	20%	0%	5%	10%	20%
Solver	Iteration								
RandomSolver	500	2.80	3.36	3.92	5.00	3.07	3.15	3.37	3.65
	1000	2.22	2.81	3.35	4.56	2.60	2.77	2.82	2.99
	2500	1.94	2.52	3.16	4.34	2.43	2.58	2.69	2.70
	5000	1.26	1.98	2.62	3.92	1.73	2.00	2.15	2.53
HillClimbing	500	2.87	3.48	4.09	5.21	3.38	3.42	3.60	3.53
	1000	3.07	3.71	4.23	5.34	3.56	3.57	3.50	3.51
	2500	2.99	3.67	4.20	5.32	3.48	3.54	3.60	3.56
	5000	2.73	3.36	3.93	4.99	3.21	3.27	3.28	3.23
TabuSearch	500	1.22	1.94	2.68	3.92	1.68	1.92	2.14	2.36
	1000	0.93	1.72	2.44	3.84	1.56	1.94	2.04	2.42
	2500	0.84	1.63	2.38	3.74	1.55	1.98	2.15	2.38

	5000	0.51	1.31	2.02	3.45	1.21	1.66	1.79	2.18
SimulatedAnnealing	500	1.76	2.51	3.10	4.32	2.14	2.35	2.43	2.57
	1000	1.18	1.94	2.61	3.86	1.78	2.06	2.17	2.55
	2500	0.72	1.55	2.16	3.56	1.31	1.67	1.80	2.24
	5000	0.47	1.24	2.00	3.32	1.05	1.59	1.70	2.19
GeneticAlgorithm	500	1.33	1.99	2.58	3.99	1.74	1.98	2.09	2.47
	1000	0.61	1.34	2.06	3.36	1.14	1.45	1.77	2.04
	2500	0.31	1.05	1.78	3.14	0.72	1.18	1.52	1.79
	5000	0.39	1.18	1.89	3.23	0.95	1.47	1.61	1.89

Tabelle D.16.: Vergleich der reaktiven Lösungsverfahren auf $m = 50$ unterschiedliche Unsicherheitsszenarien für das Instanzset m2.

Quelle: Eigene Darstellung

Instance set n0		Mean μ				Std. Dev σ			
	Uncertainty p:	0%	5%	10%	20%	0%	5%	10%	20%
Solver	Iteration								
RandomSolver	500	4.58	4.94	5.33	6.15	3.31	3.40	3.44	3.61
	1000	3.95	4.27	4.66	5.48	3.03	3.03	3.16	3.26
	2500	3.02	3.43	3.92	4.78	2.63	2.72	2.81	3.11
	5000	2.74	3.19	3.63	4.58	2.77	2.98	2.98	3.09
HillClimbing	500	3.05	3.64	4.12	5.22	3.92	4.06	4.02	4.09
	1000	3.35	3.97	4.43	5.51	4.09	4.26	4.17	4.34
	2500	3.33	3.85	4.41	5.44	4.27	4.30	4.28	4.37
	5000	2.85	3.46	4.01	5.04	3.72	3.78	3.80	3.80
TabuSearch	500	0.94	1.60	2.24	3.52	1.69	1.92	2.13	2.54
	1000	0.67	1.40	2.10	3.33	1.57	1.90	2.01	2.41
	2500	0.21	1.05	1.70	3.03	0.56	1.26	1.57	2.01
	5000	0.17	0.95	1.65	2.95	0.66	1.25	1.63	1.92
SimulatedAnnealing	500	2.14	2.76	3.26	4.35	2.80	3.06	2.95	3.15
	1000	1.06	1.78	2.34	3.50	1.73	2.10	2.26	2.68
	2500	0.48	1.19	1.86	3.18	1.13	1.47	1.73	2.11
	5000	0.33	1.06	1.75	3.04	0.77	1.23	1.45	1.99
GeneticAlgorithm	500	0.45	1.10	1.77	3.05	1.02	1.31	1.68	2.11
	1000	0.30	1.01	1.77	3.10	0.80	1.41	1.64	2.18
	2500	0.13	0.83	1.47	2.80	0.44	1.00	1.28	1.82
	5000	0.15	0.91	1.63	2.83	0.44	1.21	1.48	1.85

Tabelle D.17.: Vergleich der reaktiven Lösungsverfahren auf $m = 50$ unterschiedliche Unsicherheitsszenarien für das Instanzset n0.

Quelle: Eigene Darstellung

D.5. Weitere Vergleiche

Instance set m1		Proactive Mean μ				Predictive Mean μ				Reactive Mean μ			
	Uncertainty p:	0.00	0.05	0.10	0.20	0.00	0.05	0.10	0.20	0.00	0.05	0.10	0.20
Solver	Iteration												
RandomSolver	500	0.05	0.93	1.75	3.26	0.08	0.99	1.82	3.33	0.08	0.93	1.69	3.15
	1000	0.05	0.91	1.71	3.20	0.05	0.91	1.69	3.23	0.05	0.91	1.66	3.03
	2500	0.03	0.92	1.73	3.26	0.03	0.91	1.71	3.25	0.02	0.87	1.64	3.11
	5000	0.02	0.92	1.72	3.22	0.02	0.93	1.78	3.32	0.02	0.82	1.57	3.08
HillClimbing	500	1.78	2.56	3.33	4.78	1.80	2.57	3.33	4.79	1.88	2.63	3.34	4.67
	1000	1.85	2.65	3.39	4.84	1.81	2.58	3.31	4.78	1.82	2.48	3.27	4.70
	2500	1.80	2.60	3.36	4.83	1.76	2.54	3.30	4.74	2.09	2.83	3.48	4.92
	5000	1.84	2.62	3.38	4.85	1.83	2.59	3.36	4.77	2.04	2.78	3.50	4.83
TabuSearch	500	1.40	2.21	2.99	4.47	1.28	2.11	2.89	4.37	1.41	2.23	2.85	4.28
	1000	1.34	2.17	2.94	4.42	1.38	2.18	2.93	4.40	1.37	2.14	2.82	4.27
	2500	1.33	2.16	2.93	4.44	1.33	2.17	2.94	4.38	1.53	2.32	3.04	4.37
	5000	1.37	2.20	2.97	4.46	1.40	2.23	2.99	4.46	1.37	2.14	2.87	4.25
SimulatedAnnealing	500	1.44	2.27	3.04	4.48	1.33	2.12	2.87	4.33	1.55	2.35	3.04	4.48
	1000	1.24	2.03	2.78	4.24	1.16	1.95	2.70	4.18	1.28	2.03	2.80	4.09
	2500	0.92	1.73	2.48	3.93	0.79	1.59	2.33	3.80	0.97	1.78	2.51	3.88
	5000	0.72	1.55	2.32	3.78	0.65	1.48	2.24	3.71	0.69	1.43	2.17	3.61
GeneticAlgorithm	500	0.14	1.05	1.84	3.34	0.14	1.05	1.85	3.37	0.13	0.99	1.81	3.31
	1000	0.08	1.01	1.85	3.36	0.09	1.01	1.84	3.34	0.10	0.87	1.73	3.17
	2500	0.09	1.02	1.84	3.38	0.10	1.01	1.85	3.35	0.07	0.85	1.65	3.16
	5000	0.09	0.97	1.79	3.31	0.10	1.01	1.80	3.31	0.08	0.88	1.63	3.06

Tabelle D.18.: Vergleich der Ergebnisse der pro-, prä- und reaktiven Verfahren für das Instanzset m1

Quelle: Eigene Darstellung

Instance set m2		Proactive Mean μ				Predictive Mean μ				Reactive Mean μ			
	Uncertainty p:	0.00	0.05	0.10	0.20	0.00	0.05	0.10	0.20	0.00	0.05	0.10	0.20
Solver	Iteration												
RandomSolver	500	2.96	3.83	4.61	6.11	2.74	3.59	4.34	5.84	2.80	3.36	3.92	5.00
	1000	2.32	3.21	4.05	5.58	2.26	3.10	3.88	5.36	2.22	2.81	3.35	4.56
	2500	1.65	2.57	3.40	4.92	1.65	2.52	3.33	4.83	1.94	2.52	3.16	4.34
	5000	1.29	2.21	3.03	4.57	1.34	2.19	2.99	4.50	1.26	1.98	2.62	3.92
HillClimbing	500	2.93	3.78	4.58	6.00	2.14	2.91	3.63	5.05	2.87	3.48	4.09	5.21
	1000	3.04	3.90	4.67	6.10	2.34	3.12	3.83	5.27	3.07	3.71	4.23	5.34
	2500	2.90	3.77	4.56	5.99	2.39	3.21	3.95	5.39	2.99	3.67	4.20	5.32
	5000	2.87	3.78	4.58	6.04	2.30	3.09	3.84	5.24	2.73	3.36	3.93	4.99
TabuSearch	500	1.37	2.25	3.02	4.49	1.24	2.08	2.83	4.27	1.22	1.94	2.68	3.92
	1000	1.13	1.99	2.79	4.26	1.02	1.81	2.59	4.02	0.93	1.72	2.44	3.84
	2500	0.77	1.65	2.46	3.93	0.72	1.58	2.39	3.83	0.84	1.63	2.38	3.74
	5000	0.67	1.55	2.35	3.83	0.56	1.47	2.27	3.76	0.51	1.31	2.02	3.45
SimulatedAnnealing	500	1.89	2.74	3.50	4.99	1.72	2.49	3.23	4.67	1.76	2.51	3.10	4.32
	1000	1.19	2.08	2.90	4.35	1.16	1.94	2.68	4.11	1.18	1.94	2.61	3.86
	2500	0.74	1.60	2.38	3.83	0.81	1.63	2.38	3.85	0.72	1.55	2.16	3.56
	5000	0.50	1.41	2.24	3.67	0.48	1.36	2.16	3.63	0.47	1.24	2.00	3.32
GeneticAlgorithm	500	1.40	2.22	2.99	4.46	1.33	2.11	2.85	4.28	1.33	1.99	2.58	3.99
	1000	0.63	1.46	2.24	3.70	0.64	1.46	2.22	3.67	0.61	1.34	2.06	3.36
	2500	0.36	1.19	1.98	3.45	0.29	1.13	1.88	3.34	0.31	1.05	1.78	3.14
	5000	0.26	1.10	1.90	3.34	0.24	1.09	1.89	3.34	0.39	1.18	1.89	3.23

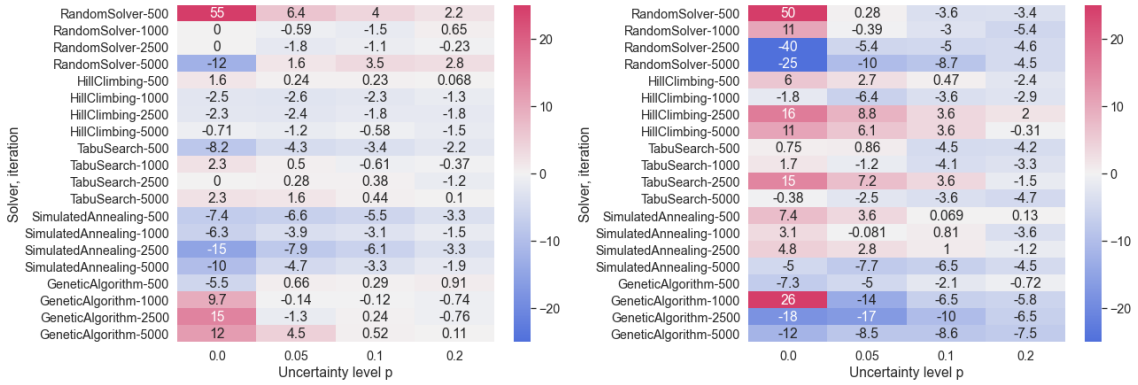
Tabelle D.19.: Vergleich der Ergebnisse der pro-, prä- und reaktiven Verfahren für das Instanzset m2

Quelle: Eigene Darstellung

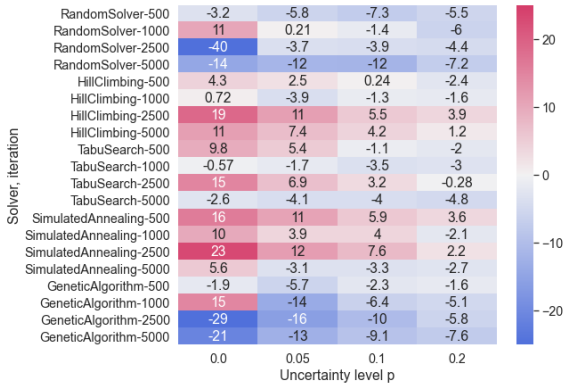
Instance set n0		Proactive Mean μ				Predictive Mean μ				Reactive Mean μ			
	Uncertainty p:	0.00	0.05	0.10	0.20	0.00	0.05	0.10	0.20	0.00	0.05	0.10	0.20
Solver	Iteration												
RandomSolver	500	4.48	5.30	6.06	7.51	4.48	5.26	5.99	7.37	4.58	4.94	5.33	6.15
	1000	3.82	4.63	5.40	6.78	3.89	4.62	5.33	6.73	3.95	4.27	4.66	5.48
	2500	3.09	3.95	4.71	6.15	3.10	3.89	4.62	5.99	3.02	3.43	3.92	4.78
	5000	2.62	3.46	4.22	5.67	2.59	3.42	4.15	5.62	2.74	3.19	3.63	4.58
HillClimbing	500	3.19	4.15	4.97	6.44	1.34	2.15	2.89	4.37	3.05	3.64	4.12	5.22
	1000	3.21	4.21	5.07	6.52	1.47	2.28	3.01	4.48	3.35	3.97	4.43	5.51
	2500	3.31	4.28	5.11	6.69	1.25	2.01	2.75	4.20	3.33	3.85	4.41	5.44
	5000	3.11	4.10	4.94	6.47	1.12	1.90	2.66	4.07	2.85	3.46	4.01	5.04
TabuSearch	500	1.01	1.94	2.71	4.22	0.72	1.55	2.29	3.72	0.94	1.60	2.24	3.52
	1000	0.57	1.47	2.26	3.70	0.39	1.22	1.98	3.44	0.67	1.40	2.10	3.33
	2500	0.24	1.17	1.97	3.44	0.28	1.06	1.80	3.20	0.21	1.05	1.70	3.03
	5000	0.12	1.07	1.88	3.33	0.10	0.93	1.66	3.11	0.17	0.95	1.65	2.95
SimulatedAnnealing	500	1.97	2.88	3.67	5.13	1.67	2.45	3.18	4.56	2.14	2.76	3.26	4.35
	1000	1.05	1.97	2.77	4.21	0.78	1.56	2.31	3.66	1.06	1.78	2.34	3.50
	2500	0.42	1.32	2.11	3.57	0.36	1.16	1.92	3.36	0.48	1.19	1.86	3.18
	5000	0.24	1.17	1.98	3.48	0.17	0.91	1.68	3.07	0.33	1.06	1.75	3.04
GeneticAlgorithm	500	0.57	1.38	2.14	3.54	0.57	1.35	2.12	3.52	0.45	1.10	1.77	3.05
	1000	0.26	1.09	1.83	3.28	0.24	0.97	1.71	3.08	0.30	1.01	1.77	3.10
	2500	0.16	0.99	1.75	3.18	0.19	0.96	1.72	3.12	0.13	0.83	1.47	2.80
	5000	0.13	0.97	1.70	3.15	0.12	0.90	1.61	3.01	0.15	0.91	1.63	2.83

Tabelle D.20.: Vergleich der Ergebnisse der pro-, prä- und reaktiven Verfahren für das Instanzset n0

Quelle: Eigene Darstellung



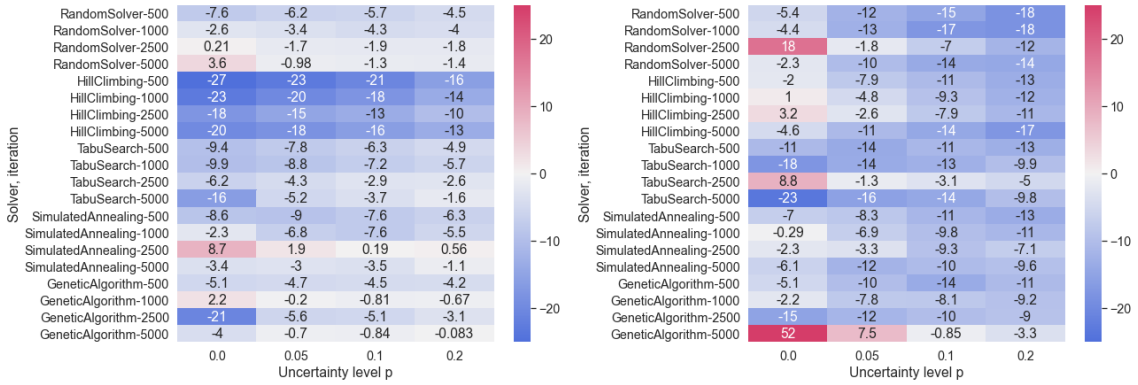
(a) Prädiktive gegenüber proaktive Methode (b) Reaktive gegenüber proaktive Methode



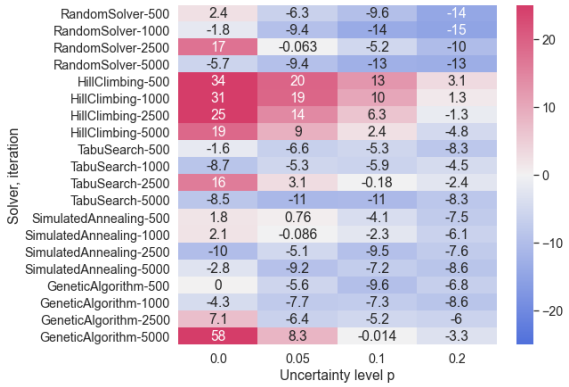
(c) Reaktive gegenüber prädiktive Methode

Abbildung D.1.: Heatmaps der prozentualen Abweichungen für das Instanzset m1

Quelle: Eigene Darstellungen



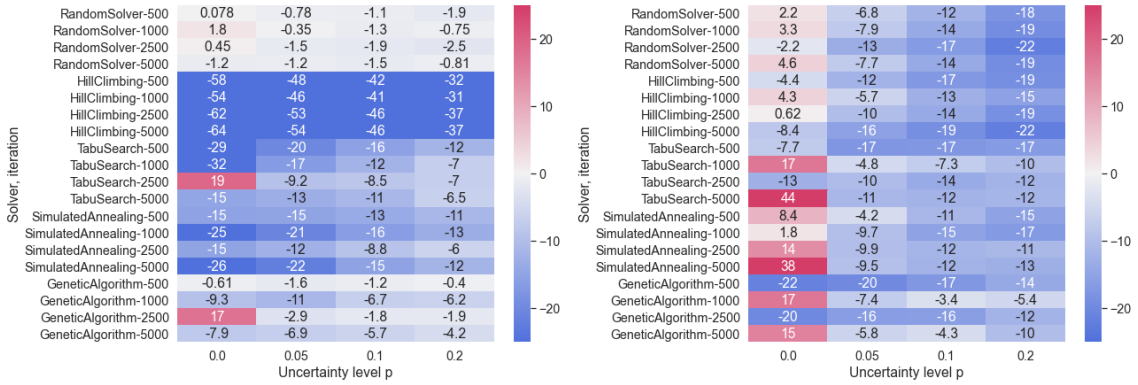
(a) Prädiktive gegenüber proaktive Methode (b) Reaktive gegenüber proaktive Methode



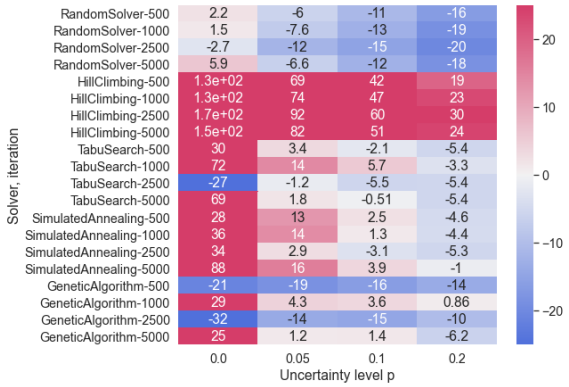
(c) Reaktive gegenüber prädiktive Methode

Abbildung D.2.: Heatmaps der prozentualen Abweichungen für das Instanzset m2

Quelle: Eigene Darstellungen



(a) Prädiktive gegenüber proaktive Methode (b) Reaktive gegenüber proaktive Methode



(c) Reaktive gegenüber prädiktive Methode

Abbildung D.3.: Heatmaps der prozentualen Abweichungen für das Instanzset n0

Quelle: Eigene Darstellungen

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.

Oldenburg, 09.03.2022

Ort, Datum

René Kuchenbuch

Unterschrift