Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik

Empirische Studie verschiedener Algorithmen für das Multi-Objective Release Planning

Diplomarbeit

zur Erlangung des akademischen Grades
Diplominformatiker

eingereicht von: Taras Iks
geboren am: 02.02.1986
geboren in: Frunse

Gutachter: Prof. Dr. rer. nat. Lars Grunske
Prof. Dr. Timo Kehrer

eingereicht am: ..... verteidigt am: .....

Zusammenfassung

In dieser Arbeit wird das Next-Release-Problem (NRP), basierend auf klassischen sowie realistischen Datensätzen, bearbeitet. Das NRP wird von fünf Algorithmen in Pareto- und Nicht-Pareto-Varianten gelöst und ausgewertet. Als Algorithmen werden NSGA-II (evolutionärer Algorithmus), Ant Colony Optimization (hybrider Algorithmus), Tabu Search und Simulated Annealing (beide lokale Suchalgorithmen) sowie Random Search verwendet. Das NRP sowie Algorithmen werden in jMetal-Framework implementiert. Weiterhin werden fünf Forschungsfragen zur Qualität der Nicht-Pareto- und Pareto-Algorithmen sowie zur Diversität, Zeit und Skalierung der Pareto-Algorithmen formuliert und anhand statistischer Daten beantwortet. Die empirische Studie hat einige interessante Entdeckungen gemacht: Obwohl Simulated Annealing die zweit schlechteste Metaheuristik für Pareto-Probleme darstellt, liefert sie für Nicht-Pareto-Probleme Lösungen auf dem Niveau vom Genetic Algorithm sowie Tabu Search und gehört somit zu den besten Metaheuristiken im Nicht-Pareto-Fall. Des Weiteren wurde festgestellt, dass Ant Colony Optimization und Random Search bei Nicht-Pareto-Problemen mit höherem Kostenfaktor bessere Lösungen im Vergleich zu anderen Metaheuristiken liefern, als wenn der Kostenfaktor geringer ist. Diese Tendenz wurde bei anderen Metaheuristiken nicht beobachtet. Bei Pareto-Problemen wurde allerdings deutlich, dass es starke Leistungsunterschiede zwischen den einzelnen Algorithmen gibt, so liefert Tabu Search in 90 % der Fälle Lösungen von besserer Qualität als alle anderen Algorithmen. Außerdem hat die Korrelationsanalyse gezeigt, dass Tabu Search gut in Bezug auf die Qualität der Lösungen skaliert.

Dankeswort

Herrn Prof. Dr. rer. nat. Lars Grunske danke ich für die Möglichkeit, diese Diplomarbeit am Lehrstuhl für Software-Engineering zu schreiben.

Herrn Prof. Dr. Timo Kehrer danke ich für seine Einwilligung, Zweitgutachter dieser Arbeit zu sein.

Ein besonderer Dank gilt M. Sc. Thomas Vogel. Seine Anregungen und konstruktive Kritik waren bei der Erstellung dieser Arbeit besonders hilfreich.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ACO	Ant Colony Optimization
Contrib	Contribution
EA	Evolutionary Algorithms
FF	Forschungsfrage
GA	Generischer Algorithmus
GD	Generational Distance
HC	Hill Climbing
HVol	Hypervolume
NFTL	No Free Lunch Theorem
NRP	Next-Release-Problem
NSGA-II	Non Dominated Sorting Algorithm II
PAP	Programmablaufplan
PF	Pareto Front
POLM	Pareto-optimale Lösungsmenge
RF	Reference Front
RP	Release Planning
RS	Random Search
SA	Simulated Annealing
SBSE	Search Based Software Engineering
TS	Tabu Search
TSP	Travelling Salesman Problem
UContrib	Unique Contribution

1 Einführung

In diesem Kapitel werden Hintergründe der in dieser Arbeit besprochenen Nicht-Pareto- und Pareto-Optimierungsprobleme beschrieben. Weiterhin werden Ziele, der Aufbau sowie die Eingrenzung der Arbeit kurz erläutert.

1.1 Motivation

Der wirtschaftliche Erfolg moderner Unternehmen wird mehr und mehr von der Möglichkeit beeinflusst, schnell und effektiv auf die veränderlichen Umstände zu reagieren. Die Unternehmen müssen ihre Produktionskapazitäten und Ressourcen in optimaler Weise einteilen, um erfolgreiche Spieler auf dem globalen Markt zu sein. Das Hauptziel der Release-Planung liegt in der Produktion vom Kunden geforderter Software auf effiziente Art und Weise. Um dieses Ziel zu erreichen, ist es heutzutage gang und gäbe, dass bei der Entwicklung die Aufgaben in Zyklen unterteilt werden. Die Programme werden nicht am Stück entwickelt, sodass alle möglichen Anforderungen umgesetzt werden, vielmehr ziehen es die Entwickler sowie Kunden vor, wenn Anforderungen im Laufe der Entwicklung verfeinert, verbessert und anhand der Kundenbedürfnisse angepasst werden. Die Schwierigkeit in dem Entwicklungszyklus entsteht bei der Festlegung der Reihenfolge der zu entwickelnden Anforderungen bzw. der Frage, welche Anforderungen in weitere Zyklen verschoben werden können. Diese Fragestellung bietet die Grundlage für das Next-Release-Problem (NRP). Die Lösung des Problems zieht alle zur Verfügung stehenden Ressourcen in Betracht und entscheidet, in Abhängigkeit von entstehenden Kosten und erzieltem Gewinn, was als Nächstes zu entwickeln ist.

Das NRP wird als NP-hartes Problem klassifiziert. NP-harte Probleme sind mit konventionellen Methoden nur schwer lösbar. In den letzten Jahrzehnten wurde eine Reihe metaheuristischer Suchalgorithmen präsentiert, um das Next-Release-Problem zu lösen. In dieser Arbeit liegt der Fokus auf folgenden Metaheuristiken: dem Genetic Algorithm (GA), der Ant Colony Optimization (ACO), der Tabu Search (TS), dem Simulated Annealing (SA) sowie der Random Search (RS). Dabei betrachtet diese Arbeit die Nicht-Pareto- als auch die Pareto-Optimierung des NRP-Problems.

Es werden fünf Forschungsfragen zur Qualität der Nicht-Pareto- als auch Pareto-Algorithmen sowie zu der Diversität, dem Laufzeitverhalten sowie der Skalierung der Pareto-Algorithmen formuliert und anhand der Experimente beantwortet. Die Experimente erfolgen anhand simulierter sowie realistischer Datensätze. Zur Auswertung der Experimente werden geeignete Metriken angewandt, um mithilfe der Resultate Schlussfolgerungen zu ziehen, wieso und unter welchen Bedingungen einige Algorithmen zur Lösung der NRP-Probleme besser geeignet sind als die anderen.

1.2 Stand der Forschung

Es gibt zahlreiche wissenschaftliche Artikeln zu Metaheuristiken. Allerdings wurde in den meisten Fällen jeweils nur eine Metaheuristik untersucht, seltener wurden mehrere Metaheuristiken miteinander verglichen [1]. Des Weiteren waren die Testdaten in vielen Fällen selbst erzeugt, sodass keine Rückschlüsse auf die Einsatzmöglichkeit der Metaheuristiken unter realen Bedingungen, z. B. in der Wirtschaft, gezogen werden konnten. Erst die Arbeiten von Xuan und Zhang [1, 2] nutzten realitätsnahe Daten, extrahiert aus öffentlichen Bug-Repositories, um unterschiedliche Algorithmen unter realen Einsatzbedingungen zu überprüfen.

Die Arbeit von Zhang zieht metaheuristische sowie hyperheuristische Versionen des Genetischen Algorithmus zusammen mit Simulated Annealing für den Vergleich auf simulierten sowie realistischen Datensätzen heran. Dabei schneiden die hyperheuristischen Techniken im Schnitt besser ab als ihre metaheuristischen Widersacher. Da die Meta- und Hyperheuristiken auf randomisierten Algorithmen basieren, nutzt Zhang statistische nichtparametrisierte Testmethoden, um auf signifikante Leistungsunterschiede zu prüfen. Die Arbeit von Xuan implementiert Backbone-Based Multilevel Algorithm (BMA) und führt eine empirische Studie durch, indem er BMA mit Simulated Annealing und dem Genetischen Algorithmus vergleicht. In der Arbeit von Xuan liegt der Fokus der BMA-Untersuchung auf NRP-Problemen mit einer Zielfunktion zur Maximierung des Gewinns. Zhang hingegen untersucht die Pareto-Variante des NRP-Problems, indem er sowohl die Maximierung des Gewinns als auch die gleichzeitige Minimierung der Kosten analysiert. Beide Arbeiten bedienen sich der gleichen Testdaten für die Experimente.

Auch andere Autoren wie Durillo et al. [3], Deb et al. [4] oder Bagnall et al. [5] können in diesem Zusammenhang genannt werden. Sie alle haben sich mit dem Problem der Release-Planung beschäftigt, allerdings nutzten sie selbst generierte Datensätze [1].

1.3 Forschungsziele

Es gibt kaum eine empirische Studie, welche unterschiedliche Algorithmen untereinander vergleicht, vor allem wenn es sich dabei um Datensätze aus der realen Welt handelt. Diese Lücke wird in der vorliegenden Arbeit geschlossen. Das grundlegende Ziel richtet sich auf die Umsetzung und Untersuchung von fünf oben genannten Algorithmen: GA, ACO, TS, SA und RS sowie ihre Erweiterung, sodass sie auf Nicht-Pareto- sowie Pareto-NRP angewandt werden können. Es erfolgt eine Evaluierung der Algorithmen anhand simulierter sowie realer Datensätze. Anschließend werden die Ergebnisse mittels Leistungsmetriken und statistischer Analysemethoden ausgewertet, sodass ein Vergleich unter den Algorithmen möglich wird. Diese Arbeit formuliert fünf Forschungsfragen zur Qualität, Diversität, Zeit und Skalierung von Pareto- sowie zur Qualität der Nicht-Pareto-Algorithmen und diskutiert am Ende die entstandenen Ergebnisse.

1.4 Wie diese Arbeit aufgebaut ist

Diese Arbeit ist wie folgt gegliedert. Kapitel 2 startet mit der Beschreibung der Search Based Software Engineering, gibt eine Motivation zur Optimierung der NP-harten Probleme und liefert eine Beschreibung unterschiedlicher Komplexitätsklassen. Des Weiteren wird in Kapitel 2 das Konzept der Metaheuristiken präsentiert. Kapitel 3 beinhaltet die Release-Planung und geht auf die Beschreibung der Eingabedaten ein. Kapitel 4 liefert die Implementierungsdetails zu den Algorithmen. Des Weiteren wird in diesem Kapitel über die Wahl der Parameter für die einzelnen Metaheuristiken gesprochen. Kapitel 5 gibt eine kurze Beschreibung des jMetal-Frameworks und bietet eine Auflistung der wichtigsten Komponenten. Das Problem der Qualitätsverifizierung wird im Kapitel 6 besprochen. Hier wird beschrieben, wie Metriken aufgebaut sind sowie interpretiert werden können. Des Weiteren gibt Kapitel 6 einen kurzen Einblick in die Methodik der statistischen Auswertung der Testdaten sowie der benutzten statistischen Methoden. Kapitel 7 beinhaltet Resultate der empirischen Studie, welche als tabellarische Vergleiche, Scatterplots sowie Boxplots dargestellt werden. Die Forschungsfragen bezüglich der empirischen Arbeit werden in Kapitel 8 formuliert, analysiert und anhand der durchgeführten Experimente beantwortet. Schließlich beendet Kapitel 9 diese Arbeit mit einem Fazit und weiteren Ausblick auf die Verbesserungsmöglichkeiten sowie offen gebliebenen Fragestellungen.

1.5 Eingrenzung der Arbeit

Diese Arbeit versucht nicht, die bestmögliche Implementierung der Algorithmen zu entwickeln sowie Leistungsunterschiede zwischen unterschiedlichen Implementierungsformen derselben Algorithmen zu messen. Das Primärziel besteht im Vergleich grundlegender Techniken anhand unterschiedlicher Datensätze. Des Weiteren versucht diese Arbeit nicht, eine komplette Sammlung möglicher Datensätze zusammenzustellen und zu analysieren. Die genutzten Metriken und statistischen Untersuchungsmethoden wurden anhand gängiger Literatur gewählt. Es gibt eine Reihe weiterer Untersuchungsmethoden, diese zu beleuchten würde den Rahmen der Arbeit überschreiten.

2 Grundlagen

Dieses Kapitel gibt einen Überblick über Search Based Software Engineering, beschreibt die Komplexität der Probleme, erläutert den Begriff der Optimierung und schildert die grundlegenden Elemente der Metaheuristiken.

2.1 Search Based Software Engineering

Search Based Software Engineering (SBSE) ist ein Forschungszweig, in welchem die suchenbasierte Optimierung auf Probleme des Software Engineering angewandt werden. Das Ziel von SBSE besteht darin, Probleme, welche sich oft als NP-hart erweisen, in Suchprobleme umzuwandeln, welche dann maschinell gelöst werden können. Ein Suchproblem ist dadurch gekennzeichnet, dass optimale oder beinah optimale Lösungen im Suchraum gesucht werden, indem die Suche von der Fitnessfunktion geleitet wird. Die Fitnessfunktion bewertet Lösungen, indem sie den Lösungen einen Wert zuweist.

SBSE hat eine klare Vorgehensweise und startet immer mit zwei grundlegenden Elementen [6]:

1. Wahl der Repräsentation eines Problems.
2. Festlegung der Fitnessfunktion.

Für viele Probleme, wie z. B. das Next-Release-Problem oder Travelling-Salesman-Problem (TSP), existieren bereits Repräsentationsformen. Bei NRP kann die Menge der Kunden und bei TSP die Menge der Städte durch einen binären oder ganzzahligen Vektor repräsentiert werden, wobei es noch weitere Einschränkungen wie z. B. Abhängigkeiten zwischen den Anforderungen eines NRP-Problems geben kann. Genauso verhält es sich mit den Fitnessfunktionen: Für NRP wird der Gewinn und für TSP die Länge der Route mithilfe der Fitnessfunktion bewertet. Sollte aber für ein bestimmtes Problem die Fitnessfunktion unbekannt sein, so können oft Metriken, welche die Lösungen des Problems evaluieren, als Ausgangspunkt zur Bestimmung der Fitnessfunktion genutzt werden. Wurde der Stand erreicht, an dem das Problem als ein Suchproblem formuliert und die Fitnessfunktion klar beschrieben ist, liegt der nächste Schritt in der Auswahl bzw. Implementierung eines Optimierungsalgorithmus. Da oft das Verhalten der Algorithmen für bestimmte Probleme unbekannt ist, bietet es sich an, mehrere Algorithmen zu nehmen, um ihre Leistung zu überprüfen. Wurden ein oder mehrere Algorithmen ausgewählt, so soll die Frage der Konfiguration der Parameter beantwortet werden. Dabei gibt es oft unterschiedliche Möglichkeiten, die Parameter einzustellen. Die Parameterwahl nimmt großen Einfluss auf das Verhalten der Algorithmen, kann aber oft nur durch eine empirische Studie bestimmt werden, da sie von vielen Faktoren wie der Probleminstanz oder Implementierung abhängt. Indem die Algorithmen auf einem Problem ausgeführt werden, lässt sich mit den Parametern experimentieren, bis ein optimales Verhalten erreicht ist. Danach werden die Resultate der Algorithmen analysiert, um anhand der Ergebnisse den Algorithmus sowie den Einfluss der Parameterwahl auf sein Verhalten zu bewerten.

Die SBSE-Vorgehensweise wird für diese Arbeit als Leitfaden genutzt. Zuerst soll aber die Frage der Komplexität der Probleme und speziell des NRP-Problems geklärt werden.

2.2 Komplexität

Die Komplexitätstheorie der Informatik befasst sich mit der Komplexität der Probleme. Sie ordnet Probleme anhand des Aufwandes, welcher zu ihrer Lösung benötigt wird, unterschiedlichen Komplexitätsklassen zu. Um die Komplexitätsklasse eines Problems zu bestimmen, muss bekannt sein, wie viel Zeit ein Algorithmus zur Lösung des Problems benötigt. Diese Zeit wird in Form elementarer Operationen gemessen. Die Komplexität wird meistens in zwei Klassen unterteilt: P und NP. Probleme, die zur Klasse P gehören, können in Polynomialzeit gelöst werden. NP-Probleme sind Probleme, welche in Polynomialzeit nicht lösbar sind.

Diese Arbeit hat nicht vor, die Klassen P und NP zu formalisieren, sondern bietet einen allgemeinen Überblick, um diese Komplexitätsklassen anhand ihrer Charakteristika und gemeinsamen Eigenschaften auf intuitiver Ebene darzustellen. Die Definition der Probleme, welche der Klasse P angehören, lautet wie folgt:

Definition 1: Klasse P

Ein Problem gehört zur Klasse P, wenn es einen Algorithmus gibt, der dieses Problem in Polynomialzeit löst [7].

NP steht dagegen für „nichtdeterministisch polynomielle Zeit“. Es repräsentiert die Klasse der Probleme, für die keine Lösung in Polynomialzeit auf deterministischer Turingmaschine berechnet werden kann, da der Suchraum eines NP-Problems exponentiell groß zur Dimension des Problems ist. Nichtdeterministisch bedeutet, dass die Maschine zu jedem Zeitpunkt potentiell mehrere Möglichkeiten hat, ihre Berechnung fortzusetzen. Anders als bei einer deterministischen Turingmaschine gibt es also keinen eindeutigen Rechenweg. Alle Probleme, die mit einer deterministischen Turingmaschine gelöst werden können, lassen sich aber auch von einer nicht-deterministischen Turingmaschine lösen. Folglich gehört Klasse P zur Klasse NP. Für NP-Probleme ist es nicht notwendig, dass die Lösung in Polynomialzeit errechnet werden kann, sondern sie muss in Polynomialzeit überprüfbar sein.

Definition 2: Klasse NP

Ein Entscheidungsproblem liegt in NP, wenn jede Lösung des Problems in Polynomialzeit überprüft werden kann [7].

Ein Entscheidungsproblem besteht dann, wenn es mit ja oder nein beantwortet werden kann. Die besonders schwer lösbaren Probleme werden als NP-hart bezeichnet.

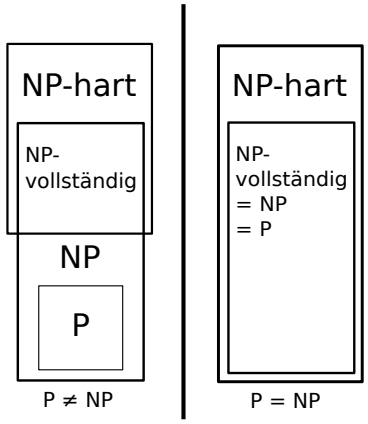


Abbildung 1: P-NP-Problem [8]

Die Klasse der NP-harten Probleme definiert, informell gesprochen, alle Probleme, welche mindestens genauso schwer lösbar sind wie jedes NP-Problem. Das bedeutet, dass sich alle anderen Probleme aus der NP-Klasse in Polynomialzeit auf das vorgegebene Problem reduzieren lassen.

Es ist anzumerken, dass es viele Probleme gibt, welche NP-hart sind, die aber nicht in NP liegen. Ein Beispiel dafür stellen Optimierungsprobleme dar, da sie optimale Lösungen suchen, was an sich kein Entscheidungsproblem ergibt und deswegen nicht zu NP gehört. Probleme, die sowohl in NP liegen als auch NP-hart sind, werden NP-vollständig genannt. Seit der Einführung des Begriffs der NP-Vollständigkeit in den 1970'er Jahren symbolisiert dieser Begriff die Komplexität mancher Probleme. Eine ganze Reihe bekannter Probleme aus dem Bereich der Mathematik, Informatik etc. sind jetzt als NP-vollständig nachgewiesen worden.

Definition 3: NP-hart

Ein Problem ist NP-hart, wenn es mindestens genauso schwer lösbar ist wie jedes Problem in NP [7].

Somit stellt sich die Frage, ob auch NP-harte Probleme effizient lösbar sind. Die Abbildung 1 beschreibt das P-NP-Problem. Zurzeit ist davon auszugehen, dass $P \neq NP$ ist, d. h., es existiert kein Algorithmus, welcher NP-harte Probleme in Polynomialzeit lösen kann. Viele Versuche, NP-Probleme in Polynomialzeit zu lösen, sind gescheitert. Sollte sich aber das Gegenteil erweisen, dann würden alle Probleme, die in NP liegen, auch zur Klasse P gehören und wären somit effizient lösbar. NP enthält zahlreiche Probleme von praktischem Interesse, wie z. B. Release-Planung, von denen noch nicht bekannt ist, ob sie in P liegen. Deswegen bleibt die Frage $P = NP$, d. h. ob auch die schwersten Probleme der Klasse NP mit deterministischen Maschinen effizient lösbar sind, eine der herausfordernden Fragen, die bis jetzt ungelöst geblieben sind.

Probleme der Release-Planung wurden als NP-hart identifiziert, sodass sogar bei kleinen

Instanzen dieser Klasse Algorithmen Schwierigkeiten bei der Berechnung aufweisen. NP-harte Release-Planungsprobleme sollen im weiteren Verlauf der Arbeit näher untersucht werden. Zuvor werden aber die Begriffe der Nicht-Pareto- sowie Pareto-Optimierung geklärt.

2.3 Optimierung

Im Bereich der Ingenieurwissenschaften ist das Lösen eines Problems oft nicht ausreichend, die gefundene Lösung muss die beste Lösung aus dem Suchraum sein. Nebenbei können bessere Lösungen im Suchraum existieren: wenn sie aber die vorgegebenen Einschränkungen des Problems nicht erfüllen, werden sie bei der Suche nicht verarbeitet.

Die Nicht-Pareto-Optimierung beinhaltet nur eine Zielfunktion und mündet normalerweise in einer einzigen optimalen Lösung, sodass ein Maximum oder Minimum eines Problems gefunden wird. Indem eine einzige Zielfunktion optimiert wird, ist nur eine partielle Sicht auf die Resultate zu gewinnen, da andere Zielfunktionen außer Acht gelassen werden. Auf der anderen Seite betrachtet die Pareto-Optimierung mehrere, oft widersprüchliche Zielfunktionen. Werden mehrere Zielfunktion gleichzeitig optimiert, so kann sich der Zielfunktionswert verschlechtern, wenn eine andere Zielfunktion verbessert werden sollte. In diesem Fall gibt es üblicherweise nicht nur eine, sondern eine Menge optimaler Lösungen. Die Pareto-Optimierung beinhaltet folgende Möglichkeiten: Minimierung aller Zielfunktionen, Maximierung aller Zielfunktionen oder Minimierung mancher und Maximierung anderer Zielfunktionen. Dabei soll beachtet werden, dass jedes Maximierungsproblem in ein Minimierungsproblem transformiert werden kann, indem der Zielfunktionswert mit -1 multipliziert wird. Die Transformation von einem Minimierungsproblem in ein Maximierungsproblem erfolgt analog. Im weiteren Verlauf der Arbeit wird nur von Maximierungsproblemen die Rede sein, wenn es die Nicht-Pareto-Optimierung betrifft. Bei Pareto-Problemen werden Maximierung des Gewinns sowie gleichzeitige Minimierung der Kosten betrachtet.

Ist ein Pareto-Problem gelöst, so entsteht als Resultat eine Menge von Lösungen. Nur eine kleine Teilmenge dieser Lösungen ist aus praktischer Sicht von Interesse. Damit eine Lösung interessant wird, darf sie von anderen Lösungen nicht dominiert werden. Ein Dominanz-Kriterium wird wie folgt formuliert:

Definition 4: Dominanz

Eine Lösung x_1 dominiert Lösung x_2, wenn x_1 mindestens genauso gut ist wie x_2 für alle Zielfunktionen, und x_1 ist strikt besser als x_2 für mindestens eine Zielfunktion [9].

Die Definition der Dominanz wird genutzt, um weitere Eigenschaften der Lösungsmenge zu beschreiben. Lösungen, welche andere Lösungen dominieren, aber selbst nicht dominiert werden, werden als Pareto-optimale oder auch nichtdominierte Lösungen bezeichnet.

Definition 5: Pareto-optimale Lösung

Eine Lösung x_1 ist Pareto-optimal, wenn keine weitere Lösung x_2 existiert, die x_1 dominiert [9].

Ein Experiment stellt die Ausführung aller Algorithmen auf einem bestimmten Datensatz dar, mit einer bestimmten Anzahl an Iterationen. Bei den Experimenten, die in den folgenden Kapiteln beschrieben werden, entstehen Pareto-optimale Lösungen in unterschiedlichen Kontexten. Bei der Ausführung des Experiments wird ein und derselbe Algorithmus mehrmals ausgeführt. In jeder Iteration entstehen Pareto-optimale Lösungen. Diese Lösungen werden dann zu einer Pareto-optimalen Lösungsmenge eines Algorithmus pro Experiment zusammengefasst, indem erneut alle dominierten Lösungen, welche bei der Zusammenfassung entstanden sind, herausgefiltert werden. Um Missverständnisse zu vermeiden und ein klares Verständnis zu schaffen, werden folgende Definitionen getroffen:

Definition 6: Pareto-optimale Lösungsmenge

Die Pareto-optimale Lösungsmenge (POLM) repräsentiert Pareto-optimale Lösungen, die von einem Algorithmus in einem Experiment gefunden werden.

Nach der Durchführung eines Experiments werden POLMs aller Algorithmen zu einer Menge zusammengefasst und es werden wieder neu entstandene dominierte Lösungen herausgefiltert. Die entstandene Menge wird in dieser Arbeit als Referenz-Front bezeichnet.

Definition 7: Referenz-Front

Die Referenz-Front (RF) definiert die Pareto-optimale Teilmenge der Vereinigung aller Lösungen, welche von der Ausführung sämtlicher Algorithmen in einem Experiment stammen [2].

In vielen Fällen ist die Information zur Pareto-Front eines Problems entweder begrenzt vorhanden oder, wie im Fall des NRP, völlig unbekannt [2]. Da aber der Entscheidungsträger meistens keine klaren Präferenzen im Voraus über das Problem besitzt, besteht das Hauptziel der Pareto-Optimierung darin, eine gute Referenz-Front zu finden. Die Referenz-Front stellt eine Menge äquivalenter Alternativen dar, die durch den Suchprozess gefunden wurden. Im NRP-Fall wird die Referenz-Front als eine Annäherung an die unbekannte Pareto-Front betrachtet.

Definition 8: Pareto-Front

Die Pareto-Front repräsentiert alle Pareto-optimalen Lösungen, welche von einem Algorithmus gefunden werden können [2].

Trotz der Existenz mehrerer Parato-optimaler Lösungen, wird in der Praxis meistens nur eine Lösung vom Entscheidungsträger ausgewählt. Da aber die Wahl einer konkreten Lösung vom Kontext abhängt, soll eine möglichst vollständige Referenz-Front berechnet werden. Um die Referenz-Front berechnen zu können, wird sich spezieller Techniken in Form von Metaheuristiken bedient. Der nächste Abschnitt gibt eine Einführung in dieses Thema.

2.4 Metaheuristiken

Dieses Kapitel beschreibt die grundlegenden Konstrukte der Metaheuristiken, die für alle Algorithmen gleich sind. Die Elemente, welche für jede Metaheuristik spezifisch sind, werden in den entsprechenden Kapiteln behandelt.

Metaheuristiken wurden Mitte der 1980er Jahre vorgestellt und sind allgemeine kombinatorische Optimierungstechniken, um NP-harte Optimierungsprobleme wie z. B. das Next-Release-Problem oder Travelling-Salesman-Problem zu lösen. Wie bereits im letzten Kapitel beschrieben, erlauben NP-harte Probleme höchstwahrscheinlich keine Berechnung exakter Lösungen in polynomialer Zeit, solange P $\neq$ NP gilt. Das bedeutet, dass nach Techniken gesucht wurde, die eine gute Annäherung in überschaubarer Zeit für Probleme dieser Art lieferten. Abhilfe dazu boten Metaheuristiken. Aufgrund der wahrscheinlichkeitsbasierten Funktionsweise sind Metaheuristiken in der Lage, NP-harte Probleme in Polynomialzeit zu lösen. Allerdings liefern sie keine Garantie, dass eine optimale Lösung tatsächlich gefunden wird. Nichtsdestotrotz können Metaheuristiken entweder eine optimale Lösung oder eine gute Annäherung an das Optimum zur Verfügung stellen. Die Annäherung kann oft als Ersatz für die optimale Lösung genutzt werden. Es muss beachtet werden, dass sich aufgrund des Zufallsprinzips der Metaheuristiken die Lösungen verschiedener Testläufe voneinander unterscheiden. Die Metaheuristiken sind eher als Konzepte und nicht als konkrete Algorithmen zu verstehen. Sie müssen für das jeweilige Problem umgesetzt werden. Zu den wichtigsten Metaheuristiken zählen Genetic Algorithm, Ant Colony Optimization, Simulated Annealing sowie Tabu Search. Random Search sowie Hill Climbing sind dagegen Heuristiken. Heuristiken können manchmal sehr schnell lokale Optima finden. Wenn das Problem allerdings darin besteht, ein globales Optimum zu ermitteln, nutzen heuristische Algorithmen wenig, da sie keine Möglichkeit haben, lokale Optima zu verlassen, und stattdessen andere Techniken, z. B. Metaheuristiken, benutzt werden sollten. Die Begriffe Heuristik, Metaheuristik sowie Hyperheuristik werden im Standardfall für einen Algorithmus benutzt, der eine optimale Lösung sucht, aber nicht sicherstellen kann, dass sie tatsächlich gefunden wird, sogar wenn eine solche existiert. Die Präposition *Meta* bei Metaheuristiken bezeichnet einen höheren Abstraktionslevel, sodass die Intensivierung und Diversifizierung in einem Algorithmus

miteinander kombiniert werden. Hyperheuristiken gehen hingegen noch eine Stufe höher und arbeiten auf der Ebene der Heuristiken.

Um den Erfolg bei der Suche nach einem Optimum zu gewährleisten, sollten bei der Ausführung des Algorithmus Lösungen gegen ein globales Optimum konvergieren. Es heißt aber nicht, dass jede neue Lösung besser sein muss als die vorhergehende. Anstatt schlechte Lösungen gleich zu verwerfen, werden diese bei Metaheuristiken beibehalten, mit der Idee, dass sie möglicherweise zu einem globalen Optimum führen könnten. In einem Problem sollte die Lösung, welche zu der Menge der erreichbaren Lösungen gehört, die Wahrscheinlichkeit $p > 0$ haben, gefunden zu werden. Wenn das Problem diese Anforderung erfüllt, dann kann jede optimale Lösung potentiell gefunden werden.

2.4.1 Fitnessfunktion

Um über die Güte der Lösungen urteilen zu können, müssen die Lösungen bewertet werden. Die Zielfunktion f bewertet die Lösungen, indem sie jeder Lösung eine reelle Zahl aus $\mathbb{R}$ zuordnet $f : M \to \mathbb{R}$. M bezeichnet dabei die Lösungsmenge. Die Fitness einer Lösung kann aus der Zielfunktion bestimmt werden. Bei einem Minimierungsproblem ist die Fitness umso höher, je kleiner der Zielfunktionswert der Lösung ist, bei Maximierungsproblemen ist es umgekehrt. Aber allgemein lässt sich sagen, je besser die Lösung in Bezug auf den Optimierungskontext, desto höher ist ihre Fitness. Die Fitnessfunktion agiert als Brücke zwischen dem Algorithmus und dem Optimierungsproblem. Metaheuristiken bewerten die Qualität der Lösungen anhand der Informationen, welche die Fitnessfunktion liefert, und nicht auf dem direkten Weg über die Struktur der Lösungen.

2.4.2 Bestimmung der Nachbarschaft

Metaheuristiken nutzen Nachbarschaftsfunktionen, um neue Lösungen im Suchraum anhand der bereits bekannten zu finden. Die Festlegung der Nachbarschaft ist dabei ein entscheidender Faktor bei der Suche. Im Fall eines kombinatorisches Optimierungsproblems mit Suchraum $\{0, 1\}^n$ wird die Nachbarschaft definiert als alle Nachbarn, die eine Hamming-Distanz k, mit $k \leq n$ von der aktuellen Lösung besitzen. Der Parameter k legt die Größe der Nachbarschaft fest, aus welcher die nächste Lösung ausgewählt wird. Wird ein kleiner Wert für k, z. B. $k = 1$ gewählt, so werden nur die direkten Nachbarn bei der Nachbarschaftssuche zur Auswahl herangezogen. Wenn k zu groß gewählt wird, im Extremfall $k = n$, stehen alle Lösungen des Suchraumes bei der Auswahl des Nachbarn mit gleicher Wahrscheinlichkeit zur Verfügung [11]. In diesem Fall wird die nächste Lösung unabhängig von der aktuellen Lösung gewählt, was das Verhalten von Random Search darstellt. In diesem Fall vernachlässigt der Algorithmus die frühere Erfahrung und die Suche kann nicht zielgerichtet in gute Bereiche des globalen Suchraumes gelenkt werden. Die Nachbarschaftsfunktion wird anhand des gegebenen Problems festgelegt. Eine einfache Nachbarschaftsfunktion kann z. B. durch einen genetischen Mutationsoperator gebildet werden.

Um das Problem der Erforschung des gesamten Suchraumes sowie die Untersuchung vielversprechender Bereiche mit einer Nachbarschaftsfunktion unter einen Hut zu bringen,

implementieren Metaheuristiken Techniken zur Diversifizierung sowie Intensivierung des Suchprozesses. Es bleibt aber oft eine offene Frage, wann und an welcher Stelle das Zufallsprinzip in das System durch Diversifizierung hineingebracht werden könnte und an welchen Stellen die Lösungen durch Intensivierung verbessert werden sollten.

2.4.3 Intensivierung/Diversifizierung

Das Problem der Ausnutzung der Intensivierung und der Diversifizierung [11] ist ein viel untersuchter Bereich der Metaheuristiken. Unter Intensivierung wird das Erforschen des Bereiches im Suchraum zum aktuellen Zeitpunkt verstanden. Auf der anderen Seite ist als Diversifizierung die Erforschung der Bereiche des Suchraumes aufzufassen, die bisher unberücksichtigt blieben. Viele Optimierungsprobleme stoßen auf das sogenannte Invensivierungs-Diversifizierungs-Dilemma (engl. exploration-explotation-dilemma). Ein hoher Grad an Intensivierung führt zu dem Verlust an Diversität (engl. diversity) der Lösungen, weil z. B. einer elitären Wahlstrategie gefolgt wird, welche dazu führt, dass der Algorithmus zu schnell konvergiert. Im schlimmsten Fall wird eine verfrühte Konvergenz zu einem lokalen Optimum ausgelöst, was das Finden eines globalen Optimums unmöglich macht. Ein zu hoher Grad der Diversifizierung führt stattdessen zu einem schlechteren Konvergenzverhalten, weil dadurch der Suchraum stärker erforscht wird.

2.4.4 No Free Lunch Theorem

An dieser Stelle könnte auf die Idee gekommen werden, einen Algorithmus zu finden, der sämtliche stochastischen Probleme besser als alle anderen Algorithmen löst. Allerdings zeigt das No Free Lunch Theorem (NFTL) [25] von Wolpert und Macready die grundlegenden Schranken eines stochastischen Suchalgorithmus. NFTL besagt, dass es kein universelles Verfahren zur Lösung von Optimierungsproblemen gibt, wenn die Menge aller Probleme betrachtet wird. Ist eine bestimmte Technik in einem Teilbereich besser als eine andere, so muss sie unausweichlich in einem anderen Bereich schlechter sein. Insbesondere sagt NFTL aus, dass kein Algorithmus, wenn er auf die Gesamtheit aller Probleme angewandt wird, besser abschneidet als ein blindes Raten. In der Praxis zeigt diese Aussage jedoch eine beschränkte Wirkung, da die Menge aller möglichen Probleme eingeschränkt ist und man es immer mit konkreten Fragestellungen wie z. B. mit NRP zu tun hat. Folglich ist es möglich, für bestimmte Problembereiche Strategien zu entwickeln, welche besser als andere abschneiden. Das hat aber zur Folge, dass ein effektiver Algorithmus aus einem Anwendungsbereich nicht unbedingt effektiv in einem anderen Kontext funktionieren wird. Um ein Beispiel dafür aus dieser Arbeit zu geben, wird an dieser Stelle vorgegriffen und in weiteren Kapiteln an konkreten Daten erläutert, dass sich SA für Nicht-Pareto-Probleme als effektiv, für Pareto-Probleme als ineffektiv erwiesen hat. Es gibt aber Metaheuristiken wie GA und TS, die stabil gute Lösungen in diesen beiden Bereichen liefern.

2.4.5 Klassifizierung

Es liegen viele Arten für die Klassifizierung der Metaheuristiken vor. Für diese Arbeit eignet sich die Klassifikation in individuums- und populationsbasierte Metaheuristiken am besten. Die erste Klasse bilden Algorithmen, welche eine lokale Suche in der Nachbarschaft durchführen, wie z. B. SA oder TS. Die zweite Klasse wird von den Metaheuristiken gebildet, die auf Populationen arbeiten, wie z. B. GA oder ACO. Eine weitere Klassifizierungsmöglichkeit besteht darin, Metaheuristiken in diejenigen einzuteilen, welche einen internen Speicher nutzen, wie z. B. TS, und diejenigen, die keinen internen Speicher gebrauchen wie z. B. SA. Tabu Search verwendet als einziger Algorithmus einen adaptiven Speicher in Form einer Tabu-Liste, um bereits besuchte Lösungen zu speichern. ACO und GA sind populationsbasiert, nutzen aber intern keinen Speicher, um Lösungen zu speichern, welche bereits verarbeitet wurden. Simulated Annealing arbeitet per Definition ohne internen Speicher.

GA	P | M
ACO	P | M
TS	1 | A
SA	1 | M
R	1 | M

Abbildung 2: Klassifizierung der Algorithmen [10]

Die Abbildung 2 beschreibt die Klassifizierung der verwendeten Metaheuristiken. Die Abkürzungen haben dabei folgende Bedeutung: A (adaptive) bezeichnet einen adaptiven Speicher, sodass der Speicher in Abhängigkeit vom Problem benutzt wird, M (memory less) steht dafür, dass der Algorithmus keinen internen Speicher nutzt, 1 repräsentiert die individuumsbasierte Verarbeitung und P steht für populationsbasierte Verfahrensweise.

Individuumsbasierte Algorithmen arbeiten auf Individuen, sodass zu Anfang ein zufälliges Individuum als Startpunkt gewählt wird. In jeder Iteration wird eine Nachbarlösung gesucht, die besser als die aktuelle ist. Individuumsbasierte Metaheuristiken nutzen eine Nachbarschaftsfunktion, um aus den direkten Nachbarn die beste Lösung auszuwählen, besitzen aber Mechanismen, um lokale Optima zu verlassen. Die populationsbasierten Metaheuristiken ziehen hingegen andere Methoden heran, um Konvergenz gegen ein lokales Optimum auszuschließen. Beim GA werden mithilfe der Mutation auch andere Bereiche des Suchraumes abgesucht, die zuvor nicht in der Population enthalten waren. Die Population bietet die Möglichkeit, einen viel größeren Teil des Suchraumes in einer Iteration abzudecken. Andererseits sind die Operationen bei individuumsbasierten Metaheuristiken viel einfacher und somit schneller. Der Vorteil von populationsbasierten Techniken liegt auch in der Möglichkeit der Parallelisierung, da jede Population oft zufällig und somit aus unterschiedlichen Bereichen des Suchraumes gebildet wird [39]. Bei individuumbasierten Techniken ist der Nutzen der Parallelisierung geringer, da diesen Techniken das Wissen aus den früheren Generationen fehlt und sie somit immer eine lokale Suche durchführen. Des Weiteren sind generationsbasierte Techniken in der Lage, gute

Charakteristika mehrerer Lösungen an die nachfolgende Generation weiterzuvererben. Der GA kombiniert Charakteristika zweier Individuen durch Kreuzung, was als hilfreich bei der Suche nach optimalen Lösungen gesehen wird. ACO nutzt hingegen Pheromone, um das Wissen in einer Population von Ameisen auszutauschen.

Eine weitere Stufe der Detaillisierung in der Klassifikation der Metaheuristiken bilden lokale, evolutionäre und hybride Suchverfahren. Lokale Suchverfahren ergeben eine Untergruppe der individuumsbasierten Verfahren. SA und TS sind individuumsbasiert und arbeiten lokal, RS ist auch indiviuumbasiert, funktioniert aber global. Die evolutionären Suchverfahren sowie hybride Suchverfahren bilden jeweils eine Untergruppe der populationsbasierten Verfahren, weil sie populationsbasiert arbeiten und dennoch relevante Unterschiede untereinander besitzen. Im Weiteren wird näher auf die konkreten Untergruppen der individuums- sowie populationsbasierten Verfahren eingegangen.

2.4.6 Lokale Suchverfahren

Lokale Suchverfahren basieren auf dem Prinzip der iterativen Verbesserung der Lösung. Ein lokales Suchverfahren betrachtet während der Ausführung im Allgemeinen nur eine einzige Lösung. Eine hohe Intensivierung des Suchprozesses wird erreicht, wenn in jeder Iteration die beste Lösung aus der Nachbarschaft ausgewählt wird. Dieses Verhalten wird als gierig (engl. greedy) bezeichnet. In lokalen Suchverfahren kommen unterschiedliche Strategien zur Überwindung lokaler Optima, im Sinne der Diversifizierung, zum Einsatz. Simulated Annealing sowie Tabu Search sind Vertreter der lokalen Suchverfahren.

2.4.7 Evolutionäre Suchverfahren

Der wichtigste Vertreter der evolutionären Suchverfahren ist der Genetische Algorithmus. Derartige Verfahren ahmen die natürliche Evolution nach, sodass die stärksten Individuen, gewertet nach ihrer Fitness, überleben. Auf die Population von Individuen werden Operatoren angewandt, sodass aus zwei oder mehreren Lösungen durch Kombination neue Lösungen entstehen. Dadurch wird im Wesentlichen die Diversifizierung in evolutionären Algorithmen erreicht. Der Hauptunterschied zwischen den evolutionären Algorithmen und der lokalen Suchalgorithmen besteht darin, dass evolutionäre Algorithmen in jedem Iterationsschritt auf einer Menge von Lösungen operieren, indem Operatoren wie Mutation oder Kreuzung auf einzelne Individuen angewandt werden. Danach entsteht eine neue Generation, indem durch die Selektion ausgewählte Individuen der Population gemäß ihrer Fitness aussortiert werden. In diesem Fall besteht die relevante Eigenschaft der evolutionären Algorithmen darin, dass sie immer eine positive Wahrscheinlichkeit haben, jede Lösung des Suchraumes im nächsten Schritt auszuwählen. Bei den lokalen Suchalgorithmen hängen die Lösungen, die im nächsten Schritt abgesucht werden, von der aktuellen Lösung ab sowie von der Nachbarschaft, welche durch die Nachbarschaftsfunktion des Algorithmus vorgegeben ist. Insbesondere im Falle einer Pareto-Optimierung sind evolutionäre Algorithmen im Vorteil, da ihre Population dazu dienen kann, einen größeren Bereich des Suchraumes abzusuchen, um eine gute Referenz-Front zu erhalten [11].

2.4.8 Hybride Suchverfahren

Ameisenalgorithmen (engl. ant colony optimization) gehören zu den hybriden Suchverfahren und kombinieren unterschiedliche Elemente der populationsbasierten sowie lokalen Suche [11]. Ameisen konstruieren Pfade, die dann von anderen Ameisen verwendet werden. Oft sind bestimmte Problembereiche den zentral gesteuerten Verfahren nicht zugänglich. Dies trifft insbesondere auf Probleme zu, bei denen Informationen dezentral vorliegen. In solchen Fällen ist eine verteilte Form der Metaheuristik angebracht, da Ameisen eigenständig handeln und Informationen mittels der Umgebung mit anderen Agenten austauschen. ACO funktioniert populationsbasiert, dennoch differenziert sie sich von evolutionären Algorithmen. Der Unterschied besteht darin, dass Ameisen bei der Entwicklung einer neuen Population lediglich die alten Lösungen verändern, sodass die bestehende Population ständig beibehalten wird. Evolutionsbasierte Techniken erzeugen hingegen in jeder Generation eine neue Population.

3 Release-Planung

Release-Planung (engl. release-planing) beinhaltet das Problem der Entscheidung der Reihenfolge für die Umsetzung der Releases. Dabei werden Anforderungen auf die jeweiligen Releases aufgeteilt, um eine Prognose über die Umsetzung erstellen zu können. Die Release-Planung kann durch einen Experten erfolgen, was einen hohen zeitlichen Aufwand, große Erfahrung und ein hohes Maß an Intuition erfordert. Aber auch in diesem Fall gäbe es kein genaues Wissen bezüglich der Qualität des erhaltenen Release-Plans. Nichtsdestotrotz kann es unter bestimmten Umständen möglich sein, den Prozess der Optimierung des Release-Plans manuell durchzuführen, was aber zu lange dauern würde sowie wirtschaftlich nicht tragbar wäre. Viele Faktoren tragen zur Veränderung der Eingabeparameter des Systems bei, was zusätzliche Korrekturen und Neuberechnungen erfordert. Unter Anderem können sich die Anforderungen über relativ kurze Zeit verändern, ihre Anzahl kann von Berechnung zu Berechnung variieren und auch Ressourcen können sich hinsichtlich Anzahl und Umfang ändern. Mit wachsender Anzahl der Ressourcen, Anforderungen, Kunden und einer ganzen Reihe von Einschränkungen erhält die Release-Planung eine hohe Komplexität, was oft eine maschinelle Berechnung erforderlich macht. Als Konsequenz davon, um einen konkurrenzfähigen und optimalen Plan bzw. eine gute Annäherung zu erhalten, muss die Berechnung mit speziellen Methoden wie z. B. Metaheuristiken erfolgen. Diese Methoden zielen darauf ab, einen robusten Release-Plan mit hoher Qualität sowie kurzer Berechnungszeit zu erstellen. Obwohl die globale Optimalität nicht garantiert werden kann, wird der Plan kontinuierlich optimiert und alle Einschränkungen werden automatisiert überprüft. Was aber in vielen Fällen von noch größerer Bedeutung ist, betrifft die Frage, ob die optimale Lösung im Vergleich zum manuellen Prozess in signifikant kürzerer Zeit erstellt werden kann. Das kann das wirtschaftliche Potential sowie die Erfolgschancen eines Produktionssystems erhöhen.

Die Release-Planung stellt eine allgemeinere Form der Planung mit einer Reihe von Release-Zyklen dar. Im nächsten Abschnitt wird auf die Spezialform der Release-Planung mit einem einzigen Release eingegangen, welche als Next-Release-Problem bekannt ist.

3.1 Next-Release-Problem

In der Anforderungsphase eines Softwareprojekts ist es ein notwendiger Schritt, eine adäquate Menge an Anforderungen für das nächste Release auszuwählen, um einen maximalen Gewinn mit begrenzten Kosten zu erzeugen. Jeder Kunde verlangt eine Teilmenge aller Anforderungen und erhält einen Gewinn, wenn diese Anforderungen im nächsten Release implementiert werden. Das NRP zielt darauf ab, eine Teilmenge der Kunden zu bestimmen, welche den maximalen Gewinn erzeugen. Die Suche einer optimalen Menge an Anforderungen, die in das nächste Release kommen, wird als Next-Release-Problem bezeichnet [13, 14, 15]. Es gibt eine Reihe von Faktoren, die bei dieser Auswahl eine Rolle spielen: Nutzer haben ihre eigenen Interessen und wollen, dass die Anforderungen, die sie für wichtig halten, in dem nächsten Release erscheinen. Aufgrund der begrenzten Ressourcen und Budgetbeschränkungen können nicht alle Anforderungen der Nutzer im nächsten Zyklus erfüllt werden. Jede Anforderung hat eigene Implementierungskosten

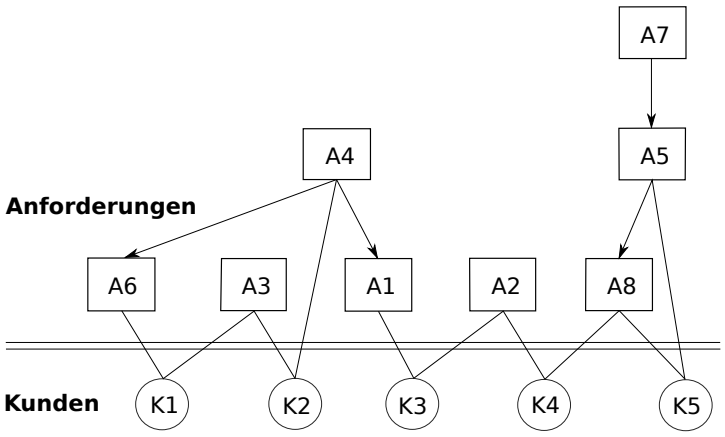


Abbildung 3: Anforderungsabhängigkeiten und Anfragen der Kunden

sowie eine unterschiedliche Priorität bei den Kunden, darüber hinaus sind auch Kunden nicht gleich wichtig für den Produzenten.

Bagnall et al. [14] haben das Next-Release-Problem beschrieben. Zhang et al. [15] schlugen eine Pareto-Formulierung für das Next-Release-Problem mit mehreren Zielfunktionen vor. Die Pareto-Formulierung mit mehreren Zielfunktionen eines Problems setzt eine Formulierung mit nur einer Zielfunktion voraus. Die Formulierung des Problems mit einer Zielfunktion und keinen Einschränkungen ist ein Spezialfall einer Pareto-Formulierung mit n Zielfunktionen, wo n = 1 ist. Des Weiteren kann ein Optimierungsproblem mit einer Zielfunktion und mehreren Einschränkungen zu einem Pareto-Problem umformuliert werden, wo die Einschränkungen durch zusätzliche Zielfunktionen abgebildet werden [2].

In den letzten Jahrzehnten wurde das NRP in vielen wissenschaftlichen Artikeln untersucht. Da es ein NP-hartes kombinatorisches Optimierungsproblem darstellt, zielt der Lösungsansatz darauf ab, eine heuristische und nicht unbedingt eine optimale Lösung zu finden, da das Erkunden des gesamten Suchraumes nicht in Polynomialzeit möglich ist. Das Ziel beim Lösen des Next-Release-Problems besteht in der Auswahl der wichtigsten Kunden und ihrer Anforderungen, sodass Kosten unterhalb des Budgets bleiben. Beim Betrachten eines NRP-Problems stehen folgende Parameter zur Verfügung:

Kunden: Eine Liste der Kunden und ihres Gewinns, der entsteht, falls alle ihre Anforderungen im nächsten Release umgesetzt werden.

Anforderungen: Liste der Kundenanforderungen, welche mit Kosten verbunden sind.

Einschränkungen: Einige Anforderungen haben Einschränkungen, indem sie Abhängigkeiten zu anderen Anforderungen aufweisen, sodass diese vor der aktuellen Anforderung implementiert werden müssen.

Budget: Das Budget ist oft begrenzt und darf nicht überschritten werden.

Da diese Arbeit auf Datensätzen aus der Arbeit von Xuan [1] basiert, so wird auch seine Definition für NRP verwendet.

In einem Software-Projekt soll die Menge aller Anforderungen durch A, mit der Kardinalität von A gleich $|A| = m$ dargestellt werden. Jede Anforderung $a_j \in A(1 \leq j \leq m)$ ist mit nicht negativen Kosten $c_j \in C$ verbunden. Ein gerichteter azyklischer Graph $G = (A, E)$ stellt die Abhängigkeiten unter den Anforderungen dar, wo A eine Menge der Knoten und E eine Menge der Kanten bezeichnen. In dem Abhängigkeitsgraphen G bezeichnet die Kante $(a_1, a_2) \in E$ eine Abhängigkeit der Anforderung a_2 von a_1,
d. h. wenn die Anforderung a_2 im nächsten Release umgesetzt werden sollte, so muss auch a_1 implementiert sein, um die Abhängigkeit zu erfüllen. Die Anforderung a_2 nennt man Kind-Abhängigkeit von a_1. Die Menge aller Anforderungen, welche a_2 über eine oder mehrere Kanten erreichen können, werden mit $parents(a_2)$ bezeichnet. Formal definiert, lautet die Definition wie folgt: $parents(a_2) = \{a_1 \in A | (a_1, a_2) \in E \text{ oder } (a_1, a_3) \in E, a_3 \in parents(a_2)\}$. Alle Anforderungen in parents($a2$) müssen implementiert sein, um Anforderung a_2 zu erfüllen.

Sei K die Menge aller Kunden, welche in Relation zu den Anforderungen A stehen, mit der Kardinalität $|K| = n$. Jeder Kunde $k_i \in S$ hat eine Menge von Anforderungen $A_i \subseteq A$. Der Gewinn des Kunden k_i soll durch $w_i \in W$ dargestellt werden. Sei $parents(A_i) = \bigcup_{a_2 \in A_i} parents(a_2)$. Für einen Kunden k_i sollen die kompletten Anforderungen durch $\tilde{A}_i = A_i \bigcup parents(A_i)$ dargestellt werden. Aufgrund der oberen Definitionen kann ein Kunde k_i zufriedengestellt werden, wenn alle Anforderungen in $\tilde{A}_i$ im nächsten Release umgesetzt werden. Die Kosten zur Zufriedenstellung des Kunden k_i sind $cost(\tilde{A}_i) = \sum_{a_j \in \tilde{A}_i} c_j$. Eine Teilmenge der Kunden $K_0 \subseteq K$ wird dabei als eine Lösung betrachtet. Man kann die Lösung auch als eine Menge der geordneten Paare darstellen, $X = \{(i, p) | p = 1, k_i \in K_0 \text{ oder } p = 0, k_i \notin K_0\}$. Die Kosten der Lösung X sind $cost(X) = cost(\bigcup_{(i,1) \in X} \tilde{A}_i)$ und die Zielfunktion von X ist $\omega(X) = \sum_{(i,1) \in X} w_i$

Definition 9: Nicht-Pareto Next-Release-Problem

Gegeben sei ein gerichteter azyklischer Graph der Abhängigkeiten $G = (A, E)$. Jeder Kunde $k_i \in K$ besitzt eine Menge der Anforderungen A_i. Der Gewinn des Kunden k_i ist $\omega_i \in W$ und die Kosten der Anforderung $a_j \in A$ werden durch $c_j \in C$ dargestellt. Das vorgegebene Budget ist b.
Das Ziel des NRP ist das Finden einer optimalen Lösung $max\ \omega(X)$ in Abhängigkeit von $costs(X) \leq b$.

Die Pareto-Definition des NRP-Problems unterscheidet sich von Nicht-Pareto-Formulierung durch eine zusätzliche Minimierung der Kosten: $min\ costs(X)$ [16].

Um alle Elemente und ihre Abhängigkeiten in einem NRP-Problem zu verdeutlichen, wird in der Abbildung 3 ein kurzes Beispiel in der Form erläutert, wie es auch bei den verwendeten Datensätzen vorliegt. Abbildung 3 präsentiert in grafischer Form die gleichen Daten, wie sie in der Abbildung 4, welche das NRP-Datenformat beschreibt, zu sehen sind. In Abbildung 3 sind fünf Kunden in Form von Kreisen, acht Anforderungen in Form von Vierecken und Abhängigkeiten zwischen den Anforderungen in Form von gerichteten Pfeilen dargestellt. So hat z. B. Kunde $K1$ zwei Anforderungen, $A6$ und $A3$. Die Anforderung $A6$ zeigt eine Abhängigkeit zu der Anforderung $A4$. Damit der Kunde

Instance group name	nrp-1	nrp-2	nrp-3	nrp-4	nrp-5
# Requirements per level	20/40/80	20/40/80/160/320	250/500/750	250/500/750/1000/750	500/500/500
Costs of requirement	1-5/2-8/5-10	1-5/2-7/3-9/4-10/5-15	1-5/2-8/5-10	1-5/2-7/3-9/4-10/5-15	1-3/2/3-5
# Maximum child requirements	8/2/0	8/6/4/2/0	8/2/0	8/6/4/2/0	4/4/0
# Requests of customer	1-5	1-5	1-5	1-5	1
# Customers	100	500	500	750	1000
# Profit of customer	10-50	10-50	10-50	10-50	10-50

Tabelle 1: Beschreibung des klassischen NRP-Datensatzes [1]

$K1$ einen Gewinn erzielen kann, sollten also Anforderungen $A3$, $A4$, $A6$ im nächsten Release umgesetzt werden. Als Beispiel betrachtet, sollen Kosten $c_1, c_2, ..., c_8$ für die Menge der Anforderungen $A = \{a_1, a_2, ..., a_8\}$ mit $7, 5, 4, 1, 2, 3, 6, 8$ dargestellt werden. Für die Menge der Kunden $K = \{k_1, k_2, ..., k_5\}$ wird der Gewinn $w_1, w_2, ..., w_5$ durch $7, 2, 6, 5, 4$ abgebildet. Wird der Kunde k_1 in der Abbildung 3 oder Abbildung 4 betrachtet, so werden seine Anforderungen durch $\tilde{A}_1 = \{a_3, a_4, a_6\}$ und die entsprechenden Kosten durch $cost(\tilde{A}_1) = 4 + 1 + 3 = 8$ dargestellt. Der Gewinn des Kunden k_1 wird durch $\omega_1 = 7$ wiedergegeben. Gegeben sei ein vordefiniertes Budget $b = 25$. Betrachtet man statt eines einzelnen Kunden eine NRP-Lösung, so kommt Folgendes zustande: Der Gewinn und die Kosten der Lösung $X_1 = \{(1, 1), (2, 0), (3, 1), (4, 0), (5, 0)\}$ betragen 7 + 6 = 13 und 7 + 5 + 4 + 1 + 3 = 20. Ähnlich dazu wird der Gewinn und die Kosten der Lösung $X_2 = \{(1, 1), (2, 0), (3, 0), (4, 0), (5, 1)\}$ berechnet, 7 + 4 = 11 und 4 + 1 + 2 + 3 + 6 + 8 = 24. Offensichtlich ist X_1 eine bessere Lösung für ein Nicht-Pareto-Problem als X_2, da sie einen höheren Gewinn unterhalb des Budgets erzielt. Die Lösung $X_3 = \{(1, 0), (2, 1), (3, 1), (4, 1), (5, 1)\}$ ist nicht valide, da diese Lösung den Gewinn von $2 + 6 + 5 + 4 = 17$ und Kosten von $7 + 5 + 4 + 1 + 2 + 6 + 8 = 33$ besitzt, sodass die Kosten das Budget übersteigen.

3.2 Beschreibung der Eingabedaten

Die Tabelle 1 zeigt die klassischen Datengruppen. Die fünf Spalten nrp-1 bis nrp-5 beschreiben fünf klassische Datensätze. Die erste Zeile gibt die Anzahl der Anforderungen pro Level in den jeweiligen Datensätzen wieder. So hat z. B. Datensatz nrp-1 20 Anforderungen im ersten, 40 im zweiten und 80 Anforderungen im dritten Level. Insgesamt besitzt Datensatz nrp-1 140 Anforderungen. Die gesamte Anzahl der Anforderungen wird später eine Rolle spielen, wenn die Skalierung der Algorithmen in Abhängigkeit von der Anzahl der Anforderungen bestimmt wird. Die zweite Zeile beschreibt die Spanne der Kosten für die Anforderungen. So liegen die Kosten beim Datensatz nrp-1 zwischen 1 bis 5 im ersten Level, zwischen 2 und 8 im zweiten und zwischen 5 und 10 im dritten. Die dritte Zeile beinhaltet die maximale Anzahl der Abhängigkeiten pro Anforderung. So haben Anforderungen im Datensatz nrp-1 höchstens acht Abhängigkeiten im Level 1, höchstens zwei Abhängigkeiten im Level 2 und keine Abhängigkeiten im Level 3. Zeile 4 beschreibt die Anzahl der Anfragen pro einzelnen Kunden. Im Datensatz nrp-1 hat der Kunde eine bis fünf Anfragen. Die Anzahl der Kunden wird in der Zeile 5 beschrieben.

Instance group name	nrp-e1	nrp-e2	nrp-e3	nrp-e4	nrp-m1	nrp-m2	nrp-m3	nrp-m4	nrp-g1	nrp-g2	nrp-g3	nrp-g4
Source repository	Eclipse				Mozilla				Gnome			
Bug report ID	150001-160000		16001-170000		200001-210000		210001-220000		450001-460000		46001-470000	
Bug reports time period	Jul.2006-Okt.2006		Okt.2006-Jan.2007		Mar.2003-Jun.2003		Jun.2003-Sept.2003		Jun.2007-Jul.2007		Jul.2007-Aug.2007	
# Requests of customer	4-20	5-30	4-15	5-20	4-20	5-30	4-15	5-20	4-20	5-30	4-15	5-20
# Requirements	3502	4254	2844	3186	4060	4368	3566	3643	2690	2650	2512	2246
# Cost of requirement	1-7	1-7	1-7	1-7	1-7	1-7	1-7	1-7	1-7	1-7	1-7	1-7
# Customers	536	491	456	399	768	617	765	568	445	315	423	294
# Profit of customer	10-50	10-50	10-50	10-50	10-50	10-50	10-50	10-50	10-50	10-50	10-50	10-50

Tabelle 2: Beschreibung des realistischen NRP-Datensatzes [1]

Der Datensatz nrp-1 umfasst genau 100 Kunden. In der letzten Zeile wird der Gewinn der Kunden beschrieben, sodass ein Kunde beim Datensatz nrp-1 einen Gewinn zwischen zehn und 50 Einheiten bekommt, wenn alle seine Anforderungen umgesetzt werden.

Die Daten entstammen der Arbeit von Xuan [1]. Laut Xuan wurden die klassischen Daten aus der NRP-Literatur abgeleitet, die realistischen Daten wurden aus drei Software-Projekten extrahiert. Dazu wurden Bug-Repositories des Eclipse-, Mozilla- sowie des Gnome-Projekts ausgewertet. Diese Daten können als pseudo-real betrachtet werden, da sie aus realen Projekten extrahiert wurden. Ob diese Anforderungen die tatsächlichen Anforderungen der realen Welt widerspiegeln, steht für die Diskussion offen. Diese Datensätze wurden bereits in anderen wissenschaftlichen Arbeiten, z. B. von Zhang [2], bedingt durch den Mangel an Datensätzen aus der realen Wirtschaft, benutzt.

Die Tabelle 2 mit realistischen Datengruppen ist analog zu Tabelle 1 zu deuten, mit einigen Anmerkungen. Die realistischen Daten sind in drei Gruppen gegliedert. Der Datensatz-Name wird genauso wie bei klassischen Daten zusammengesetzt, dazu kommt aber noch einer der Buchstaben e, m oder g, welcher für: Eclipse, Mozilla oder Gnome steht. Die realistischen Datensätze haben im Vergleich zu den klassischen Datensätzen keine Abhängigkeiten zwischen den Anforderungen. So besitzt bspw. der Datensatz nrp-e1 3502 Anforderungen bei 536 Kunden, aber keine Abhängigkeiten.

3.2.1 Format der NRP-Daten

Die Abbildung 4 wurde aus der Abbildung 3 abgeleitet und mit fiktiven Kosten der Anforderungen und dem Gewinn der Kunden ergänzt, um anhand eines Beispiels die Struktur der klassischen sowie realistischen Datensätze zu zeigen. Die Datensätze für Experimente sind als Text-Dateien gespeichert. Es sind 17 Dateien mit fünf klassischen und zwölf realistischen Datensätzen. Jede Datei verfügt dabei über eine Struktur, wie sie in der Abbildung 4 zu sehen ist. Sie wird im Folgenden Zeile für Zeile erläutert.

Die erste Zeile beschriebt den Level der Anforderungen in dem gegebenen Datensatz. In diesem Beispiel ist der Level der Anforderungen gleich 3. Die zweite Zeile gibt die Anzahl der Anforderungen im Level 1 wieder. Die dritte Zeile beinhaltet die Kosten der Anforderungen in Level 1. Da das Beispiel nur eine Anforderung im ersten Level besitzt, ist die Anzahl der Kosten auch gleich eins und beträgt sieben Einheiten. Dabei bezeichnet der Index in dieser Zeile die Nummer der Anforderung und der Wert die Kosten, bspw. costs(1) = 7. Die Zeile vier beschreibt die Anzahl der Anforderungen im Level 2 und ist

im Beispiel gleich 2. Die nächste Zeile, Zeile fünf, legt die Kosten der Anforderungen im Level 2 dar. So hat die erste Anforderung die Kosten gleich 5 und die zweite gleich 4. Der Index 1 präsentiert die erste Anforderung, der Index 2 die zweite usw., d. h. costs(1) = 5, costs(2) = 4. Die Zeile 6 beinhaltet die Anzahl der Anforderungen im Level 3 und ist gleich 5. Die Zeile 7 beschreibt die Kosten der Anforderungen im Level 3 und beträgt: costs(1) = 1, costs(2) = 2, costs(3) = 3, costs(4) = 6 und costs(5) = 8. Die nächste Zeile, Zeile 8, beschreibt die gesamte Anzahl der Abhängigkeiten zwischen den Anforderungen in allen Levels. Die Abhängigkeiten sind entsprechend der Abbildung 3 gebildet. Hierbei hat der Level 1, wie in den Tabellen 1 und 2 beschrieben, im klassischen Datensatz keine Abhängigkeiten und der realistische Datensatz besitzt grundsätzlich keine Abhängigkeiten zwischen den Anforderungen. Aus diesem Grund beschreibt dieses Beispiel die Struktur eines klassischen Datensatzes. In dem dargelegten Beispiel existieren vier Abhängigkeiten. Die Zeilen neun bis zwölf geben die Abhängigkeiten wieder. So beschreibt die Zeile 9, dass die Anforderung 6 eine Abhängigkeit zur Anforderung 4 aufweist. Es ist wichtig zu beachten, dass die Abhängigkeiten transitiv sind, d. h., wenn Anforderung A von der Anforderung B abhängt und Anforderung B von Anforderung C, so hängt auch A von C ab. Die Zeile 13 beschreibt die Anzahl der Kunden und ist gleich 5. Die Zeilen 14 bis 19 stellen den Gewinn des Kunden mit der ersten Ziffer dar, die Anzahl der Anforderungen des Kunden mit der zweiten Ziffer, danach kommt die Anforderungsliste mit den Indexen der Anforderungen. Konkret beschreibt die Zeile 14 Folgendes: Der Kunde Nummer 7 besitzt zwei Anforderungen mit den Nummern 6 und 3. Die weiteren Zeilen sind analog zu deuten.

```
3 <-------- Level der Anforderungen
1 <-------- Anzahl der Anforderungen in Level 1
7 <-------- Kosten der Anforderungen in Level 1
2 <-------- Anzahl der Anforderungen in Level 2
5 4 <-------- Kosten der Anforderungen in Level 2
5 <-------- Anzahl der Anforderungen in Level 3
1 2 3 6 8 <-------- Kosten der Anforderungen in Level 3
4 <-------- Anzahl der Abhängigkeiten
6 4 <-------- ID der Abhängigkeit A1    ID der Abhängigkeite B1
1 4 <-------- ID der Abhängigkeit A2    ID der Abhängigkeite B2
8 5
5 7 <-------- ID der Abhängigkeit A4    ID der Abhängigkeite B4
5 <-------- Anzahl der Kunden
7 2 6 3 <-------- Gewinn des Kunden 1 (7)    Anzahl der Anforderungen (2)    Anforderungsliste  (6, 3)
2 2 3 4
6 2 1 2
5 2 2 8
4 2 8 5 <-------- Gewinn des Kunden 5 (4)    Anzahl der Anforderungen (2)    Anforderungsliste  (8, 5)
```

Abbildung 4: Beschreibung des Formats der NRP-Daten

4 Implementierung der Algorithmen

Dieses Kapitel bietet einen Überblick über die Konzepte, Struktur sowie Implementierung der genutzten Algorithmen. Es beginnt mit der Beschreibung des Genetischen Algorithmus, weil seine Operatoren auch Verwendung in anderen Algorithmen finden. Weiterhin folgt die Beschreibung von ACO, RS, SA sowie TS. Zum Schluss des Kapitels wird über die Parameterwahl der Algorithmen diskutiert.

4.1 Genetic Algorithm

Evolutionäre Algorithmen [17, 18] bilden eine Klasse stochastischer Optimierungsverfahren, deren Funktionsweise der Evolution der Lebewesen entspricht. Sie werden vorrangig zur Optimierung oder Suche eingesetzt. Evolutionäre Algorithmen sind Metaheuristiken, sodass sie keine Verbesserung garantieren können, allerdings mündet ihre Suche oft in einem globalen Optimum. Der Genetische Algorithmus ist ein Vertreter dieser Klasse. Jede einzelne Lösung wird im Genetischen Algorithmus durch ein Individuum dargestellt, welches in einer Population verarbeitet wird. GA nutzt hauptsächlich drei Operatoren: Mutation, Selektion und Kreuzung. Diese Operatoren werden auf ein oder mehrere Individuen angewandt. Als Erstes soll aber die Frage der Terminologie geklärt werden:

- **Individuum**: Es wird unter einem Individuum ein möglicher Lösungskandidat eines zu optimierenden Problems verstanden. Dabei besteht ein Individuum in der Regel aus einer Zeichenkette der gewählten Repräsentationsform. Diese kann je nach Codierung als binärer oder ganzzahliger Vektor einer festgelegten Länge dargestellt werden. Die Darstellungsform des Individuums ist für das zu optimierende Problem von Bedeutung und muss deshalb problemspezifisch gewählt werden.

- **Population**: Eine Population stellt die gesamte Menge aller potentiellen Lösungskandidaten für ein Problem dar. Genetische Algorithmen arbeiten stets auf einer ganzen Population und nicht auf einzelnen Individuen.

- **Generation**: Mit Generation wird eine Population zu einem bestimmten Zeitpunkt innerhalb eines Genetischen Algorithmus bezeichnet. Jeder Iterationsschritt innerhalb eines Genetischen Algorithmus entspricht einer Generation. Durch die sukzessive Anwendung von Operatoren wird in jedem Schritt eine Generation durch eine neue ersetzt.

- **Fitness** beschreibt das Maß der Adaptierung eines Lösungskandidaten bezüglich des zu optimierenden Problems. Sie wird mittels der gewählten Fitnessfunktion für das jeweilige Individuum ermittelt.

- **Operatoren**: Genetische Operatoren werden auf Individuen einer Population angewandt, um eine Verbesserung der Lösung zu erhalten.

In der Welt der Genetischen Algorithmen bilden Individuen, welche der Evolution unterworfen sind, Lösungen in dem Suchraum eines Optimierungsproblems. Die Menge an Lösungen in einer Population, die simultan im Genetischen Algorithmus verarbeitet wird, wird in jeder Iteration verbessert, bis eine Abbruchbedingung erreicht ist. In jeder Generation werden Individuen reproduziert, überleben oder verschwinden aus der Population unter Anwendung eines Selektionsoperators. Selektion entscheidet, ob und wie viele Male ein Individuum in der nächsten Generation reproduziert wird. Allerdings muss sichergestellt sein, dass der gesamte Algorithmus eine Tendenz besitzt, Populationen mit immer besseren Individuen zu bilden, damit die Konvergenz gegen ein globales Optimum gewährleistet ist. Der Genetische Algorithmus produziert iterativ aufeinander folgende Generationen, wobei die Größe einer Population ständig konstant gehalten wird. Über die Generationen bleibt das Ziel in der Verbesserung des Fitnesswertes von Individuen erhalten. Am Anfang der Optimierung benötigt GA eine Menge an initialen Lösungen. Eine Möglichkeit besteht darin, eine per Zufallsprinzip erzeugte Menge an Lösungen zu erstellen. Diese Variante wird bei Problemen bevorzugt, wo kein Wissen über den Suchraum des Problems existiert. Im Fall eines NRP hat die praktische Erfahrung gezeigt, dass die Erstellung der initialen Population für Nicht-Pareto- und Pareto-Probleme Unterschiede aufweist. Im Nicht-Pareto-Fall ist es möglich, eine zufällige initiale Population zu erstellen, welche eine größtmögliche Diversität des genetischen Materials enthält, damit der Suchraum maximal abgedeckt ist. Das hat zur Folge, dass der Algorithmus sehr schnell gegen ein globales Optimum konvergiert. Diese Vorgehensweise führt aber bei Pareto-Problemen dazu, dass bestimmte Bereiche des Suchraumes nicht untersucht werden, z. B. Bereiche nahe bei 0. Um das Problem zu umgehen und eine größtmögliche Diversität der POLM zu gewährleisten, wird eine initiale Population bei Pareto-Problemen aus Individuen erstellt, die eine Fitness nahe bei 0 aufweisen. Dadurch wird es möglich, die Bereiche des Suchraumes nahe am Ursprung zu erforschen.

4.1.1 Genetische Operatoren

Im Folgenden wird auf die drei genetischen Operatoren Selektion, Kreuzung und Mutation eingegangen. Der Selektionsoperator bestimmt Individuen, die an der Reproduktion teilnehmen. Sie werden in Abhängigkeit von ihrer Fitness möglicherweise sogar mehrmals gewählt. Die Selektion hat einen starken Einfluss auf die Lenkung der Suche in vielversprechenden Bereiche, um gute Lösungen in kurzer Zeit zu finden.

Die Kreuzung zweier Individuen macht die Vererbung genetischen Materials an weitere Generationen möglich. Der Kreuzungsoperator lässt GA, im Vergleich zu anderen Algorithmen, zu etwas Besonderem werden. Er wird benutzt, um zwei neue Individuen aus den bereits existierenden Individuen zu erschaffen. Die Ein-Punkt-Kreuzung (engl. single-point-crossover) stellt die einfachste Möglichkeit der Kreuzung dar, indem zwei Individuen an einer zufälligen Stelle geschnitten werden, sodass die Teillösungen, welche zuvor Teile der Eltern waren, ausgetauscht werden. Der Erfolg der Kreuzung hängt von der Beschaffenheit des Suchraumes ab. Je mehr lokale Optima im Suchraum vorhanden sind, desto wahrscheinlicher erzeugt die Kreuzung zweier Individuen, die sich in einem lokalen Optimum befinden, einen Nachfahren in einem Tal dazwischen [17]. In der Praxis stellt

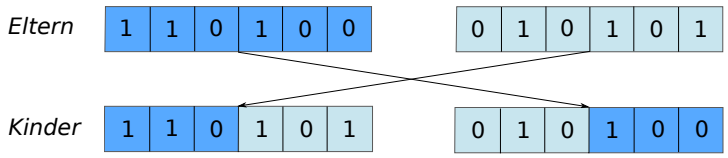


Abbildung 5: Genetischer Kreuzungsoperator

der Kreuzungsoperator das grundlegende Element des Algorithmus dar, wohingegen die Mutation eine viel geringere Rolle spielt. Die Kreuzung versucht die vorteilhaften Aspekte der Kandidat-Lösungen zu bewahren und unerwünschte Komponenten zu eliminieren. Die zufällige Funktionsweise der Mutation besitzt dagegen eine höhere Wahrscheinlichkeit, eine gute Lösung zu verschlechtern, als eine schlechte zu verbessern, weil davon auszugehen ist, dass es verhältnismäßig nur wenige gute Lösungen im Suchraum gibt [11]. Indem die Kreuzung der schwachen Kandidaten eingeschränkt wird, eliminiert der Genetische Algorithmus nicht nur diese schwachen Lösungen, sondern auch alle potentiellen Nachfolger dieser Lösungen. Diese Tatsache ermöglicht dem Algorithmus, gegen Lösungen mit höherer Qualität innerhalb von nur wenigen Generationen zu konvergieren. Die Abbildung 5 beschreibt die Funktionsweise des Kreuzungsoperators, wie er in dem Genetischen Algorithmus verwendet wird. Es werden zwei Lösungen genommen, welche als Eltern bezeichnet werden. Danach wird per Zufall ein Schnittpunkt festgelegt. In der Abbildung 5 liegt der Schnittpunkt bei drei. Jede Lösung wird dann an dieser Stelle geschnitten und durch die Hälfte der anderen Lösung ersetzt, sodass am Ende zwei neue Lösungen entstehen, welche das genetische Material der Eltern zur Hälfte geerbt haben.

Die Mutation erlaubt dem Algorithmus, weitere Bereiche des Suchraumes zu erforschen, die zuvor nicht in der Population enthalten waren. Während der Verarbeitung durch den Mutationsoperator werden Individuen einer Population, gemäß der festgelegten Rate geändert. Ist die Rate zu hoch gesetzt, ähnelt der Algorithmus einer Random Search. Im Vergleich zu der Kreuzung, wird bei der Mutation nur ein Kind aus nur einem Elternteil produziert. Letztendlich soll dieser Operator helfen, die verfrühte Konvergenz zu vermeiden, um ein globales Optimum zu finden. Die Abbildung 6 zeigt vier Mutationsoperatoren:

- **Flip**: Die Flip-Mutation wählt einen zufälligen Bit in der Ausgangslösung und kippt ihn.
- **Exchange**: Die Exchange-Mutation wählt zwei zufällige Bits in der Ausgangslösung und tauscht sie aus.
- **Insert**: Die Insert-Mutation wählt eine zufällige Position in der Ausgangslösung und fügt dort einen zufälligen Bit hinzu.
- **Remove**: Die Remove-Mutation löscht einen zufälligen Bit aus der Ausgangslösung.

Der Mutationsoperator ist in dieser Arbeit vom besonderem Interesse, da seine leichte Abwandlung in Form eines FlipOrExchange-Operators in der Nachbarschaftsfunktion bei der Entwicklung von GA, TS, SA und RS eingesetzt wurde.

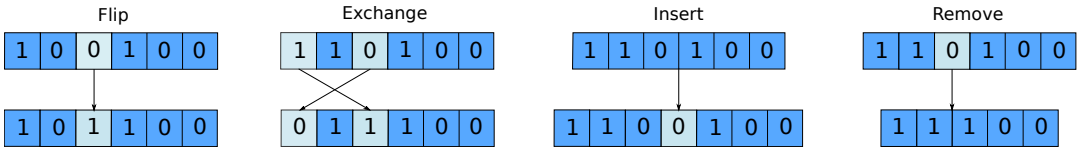


Abbildung 6: Genetische Mutationsoperatoren [17]

4.1.2 FlipOrExchange-Mutation

Bevor in weiteren Abschnitten die lokalen Suchalgorithmen im Detail beschrieben werden, sollte an dieser Stelle die Nachbarschaftsfunktion, welche extra dafür entwickelt worden ist, diskutiert werden. Die Nachbarschaftsfunktion für individuumsbasierte Suchalgorithmen wird im Kontext dieser Arbeit durch FlipOrExchange-Operator gebildet und beinhaltet zwei genetische Operatoren: Flip-Mutation und Exchange-Mutation, die mit einer festgelegten Wahrscheinlichkeit die Lösung verändern. Im Laufe der Arbeit wurde festgestellt, dass der Flip-Operator allein gut für eine schnelle Konvergenz ist und bei Nicht-Pareto-Problemen eingesetzt werden kann, da dort nur ein Maximum gesucht wird, sodass die Schritte, welche zum Maximum führen, irrelevant sind und grobkörnig gewählt sein können. Aber auch da erwies sich der Flip-Operator als zu gierig, wenn an die Budgetgrenze gestoßen wird, sodass dieser Operator schnell zu Lösungen führt, die das vorgegebene Budget überschreiten, sodass nicht valide Lösungen generiert werden. Eine größere Bedeutung hat dieses Phänomen bei Pareto-Problemen, wo die schnelle Konvergenz dazu führt, dass oft keine optimale Lösung aus dem durchsuchten Bereich gefunden wird, was zur schlechten Qualität der POLM führt. Um das Problem der schnellen Konvergenz zu lösen, wurde der Exchange-Operator benutzt, welcher keinen neuen Bit in der Lösung setzt, sondern einen Austausch der bereits vorhandenen Bits durchführt, sodass die Wahrscheinlichkeit der Erzeugung nicht valider Lösungen abnimmt. Allerdings lässt sich auf den Flip-Operator auch nicht komplett verzichten, da er für die Diversifizierung sorgt, was dem Algorithmus ermöglicht, neue Bereiche des Suchraumes zu erschließen. Der FlipOrExchange-Operator ist so aufgebaut, dass in jeder Iteration entweder der Flip- oder der Exchange-Operator verwendet wird. Der FlipOrExchange-Operator hat im Vergleich zu den einfachen genetischen Operatoren eine bessere Leistung gezeigt und wurde aus diesem Grund für alle individuumsbasierten Metaheuristiken in der Nachbarschaftsfunktion sowie bei GA als Mutationsoperator eingesetzt.

Eine weitere detaillierte Untersuchung des Einflusses unterschiedlicher Mutationsoperatoren übersteigt aber die Grenzen dieser Arbeit.

4.1.3 Programmablaufplan

Die Abbildung 7 liefert die Beschreibung des Genetischen Algorthmus, wie er in jMetal implementiert ist, wobei die grundlegende Struktur für den Genetischen Algorithmus mit einer Zielfunktion und NSGA-II mit mehreren Zielfunktionen identisch ist. Den Unterschied bilden die verwendeten Operatoren, sodass NSGA-II z. B. eine rangbasierte Sortierung durchführt.

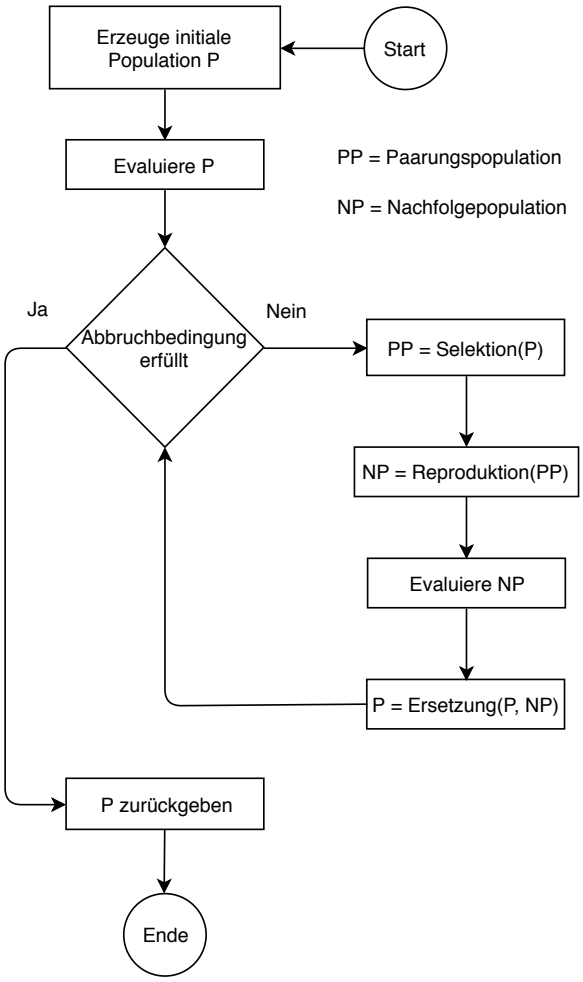


Abbildung 7: Nicht-Pareto Genetic Algorithm PAP

1. Im ersten Schritt wird die initiale Population erzeugt. Die Initialisierungsprozedur wurde so verändert, dass die initiale Population Individuen mit Fitnesswerten nahe bei 0 beinhaltet, weil andernfalls der Algorithmus zu schnell konvergiert und der Bereich am Ursprung unerforscht bleibt.

2. Im zweiten Schritt wird mithilfe des Selektionsoperators aus der aktuellen Population zufällig eine Paarungspopulation ausgewählt, die dann bei der Reproduktion eingesetzt wird.

3. Im dritten Schritt erfolgt die Reproduktion, welche die Mutation und Kreuzung von jeweils zwei Individuen aus der Paarungspopulation beinhaltet.

4. Im vierten Schritt erfolgt die Ersetzung, sodass aus zwei Populationen der Größe |2P| eine neue Population der Größe |P| entsteht, indem nur die besten Individuen beider Populationen überleben.

5. Falls die Abbruchbedingung nicht erfüllt wurde, wird die Schleife ab (2) wiederholt.

Im folgenden Abschnitt wird eine Pareto-Variante des Genetischen Algorithmus NSGA-II präsentiert, die zur Lösung der Pareto-NRP-Probleme eingesetzt wurde und in der Lage ist, eine POLM zu einem Pareto-Problem zu berechnen.

4.1.4 NSGA-II

NSGA-II steht für Non-dominated Sorting Genetic Algorithm II [19] und ist ein Genetischer Algorithmus für Probleme mit mehreren Zielfunktionen. Deb et al. präsentierten diesen elitären evolutionären Algorithmus und schlugen vor, eine Population in jeder Generation neben der initialen Population beizubehalten. Diese externe Population nimmt an allen Operationen teil. In jeder Iteration wird eine kombinierte Population der Größe |2P| aus der aktuellen und externen erstellt. Der NSGA-II-Algorithmus beinhaltet eine Pareto-optimale Sortierungsprozedur, welche auf Rängen basiert. Die Lösungen erhalten einen Rang in Abhängigkeit von dem Dominanz-Kriterium, z. B. bekommen alle nichtdominierten Lösungen den Rang 0. Der Rang der Lösung dient als primäres Sortierkriterium. Des Weiteren misst der Algorithmus die Crowding Distance der Lösungen, um eine größtmögliche Diversität der Lösungen zu gewährleisten. Die Diversität der Lösungen bildet das sekundäre Sortierkriterium. Schließlich, nutzt der NSGA-II-Algorithmus eine elitäre Selektionsstrategie, welche die Konvergenz gegen das globale Optimum beschleunigt.

Wie in Abbildung 7, welche die Struktur für den einfachen GA als auch für NSGA-II darstellt, zu sehen, wird im ersten Schritt die Selektion ausgeführt. Die Individuen werden mittels Binary Tournament Selection ausgewählt. Dabei werden immer zwei zufällige Lösungen der Population miteinander verglichen, sodass stets die bessere Lösung in die Paarungspopulation gelangt. Ist die Paarungspopulation vollständig gefüllt, so kommt es zu der zweiten Phase – der Reproduktion. Die Reproduktion erfolgt mithilfe der Ein-Punkt-Kreuzung. Es werden mit dem Kreuzungsoperator aus jeweils zwei Eltern zwei Kinder erzeugt. Die Kinder werden mit dem FlipOrExchange-Operator mutiert und

zur Nachfolgepopulation hinzugefügt, bis die Population ihre volle Größe erreicht hat. In der dritten Phase wird die Nachfolgepopulation evaluiert. In der vierten Phase der Ersetzung werden zwei Populationen, die Elternpopulation (P) sowie Nachfolgepopulation (NP), vereinigt, sodass eine rangbasierte Selektion durchgeführt werden kann. Dazu wird die gemeinsame Population der Lösungen mit Pareto-dominierter Sortierung in unterschiedliche Fronten mit dem jeweiligen Rang partitioniert. Die Selektion startet mit den Lösungen der ersten Front mit Rang 0 und fährt aufsteigt fort, bis die normale Populationsgröße wieder erreicht ist. Wenn die Front nicht vollständig in die Population hinzugefügt werden kann, dann erfolgt eine Sortierung nach Crowding Distance, sodass Individuen, welche größte Diversität mitbringen, in die Population hinzugefügt werden.

Der NSGA-II-Algorithmus wurde weitestgehend in seiner Basisform belassen. Die einzige Änderung, die vorgenommen wurde, betrifft die Erweiterung um einen Reparatur-Operator, der nach der Reproduktion eingefügt wurde, um alle nicht validen Lösungen, welche durch die Mutation in der Population entstanden sind, zu entfernen und die Population durch neue valide Lösungen bis zur vollen Größe wieder aufzufüllen.

4.2 Ant Colony Optimization

Die Ameisenalgorithmen [22, 23] gehören zu den metaheuristischen Verfahren der kombinatorischen Optimierung, die das modellhafte Verhalten von realen Ameisen bei der Futtersuche abbilden. Dieser Ansatz wurde von Dorigo et al. [24] vorgeschlagen und simuliert die kollektive Fähigkeit der Ameisen, bestimmte Probleme zu lösen, die bei Ameisenkolonien beobachtet wurden. Die einzelnen Mitglieder der Kolonie sind individuell in ihren Fähigkeiten sehr eingeschränkt. Die Kolonie als Ganzes kann hingegen komplexe Aufgaben bewerkstelligen. In der Natur geht es insbesondere um die Suche eines kürzesten Pfades zwischen dem Ameisenhaufen und der Futterquelle. Es wurde festgestellt, dass Ameisen letztendlich immer demselben Pfad zur Futterstelle folgen, welcher sich als der kürzeste von allen erweist.

In der SBSE wird diese Fähigkeit modelliert, um kombinatorische Optimierungsprobleme zu lösen. Um den ACO-Algorithmus bei der Optimierung benutzen zu können, wird das Problem in eine spezielle Form transformiert, sodass ein optimaler Pfad in einem gewichteten Graphen gesucht wird. Dabei werden Kunden eines NRP-Problems durch Knoten des Graphen dargestellt. Die künstlichen Ameisen bauen inkrementell Lösungen im Graphen auf, indem sie durch den Graph wandeln und Pheromone auf den zurückgelegten Kanten hinterlassen. Die Konstruktion des Pfades ist ein stochastischer Optimierungsprozess, basierend auf der Distanz zum nächsten Knoten und dem Pheromonlevel auf den Kanten. Die Pheromonwerte auf den Kanten werden zur Laufzeit von Ameisen modifiziert. Alle Mitglieder der Kolonie nehmen Pheromone wahr und haben eine Tendenz, den Pfaden mit höheren Pheromonwerten zu folgen. Durch die Pheromone werden Artgenossen über wichtige Ereignisse, z. B. Futterstellen, informiert.

Die Fähigkeit, den kürzesten Pfad zu finden, ist das Resultat einer Kommunikation durch die Umgebung, welche Stigmergie genannt wird. Bei der Stigmergie handelt es sich um eine indirekte, asynchrone und nicht symbolische Form der Kommunikation. Die Ameisen tauschen Informationen aus, indem sie ihre Umgebung verändern. Die

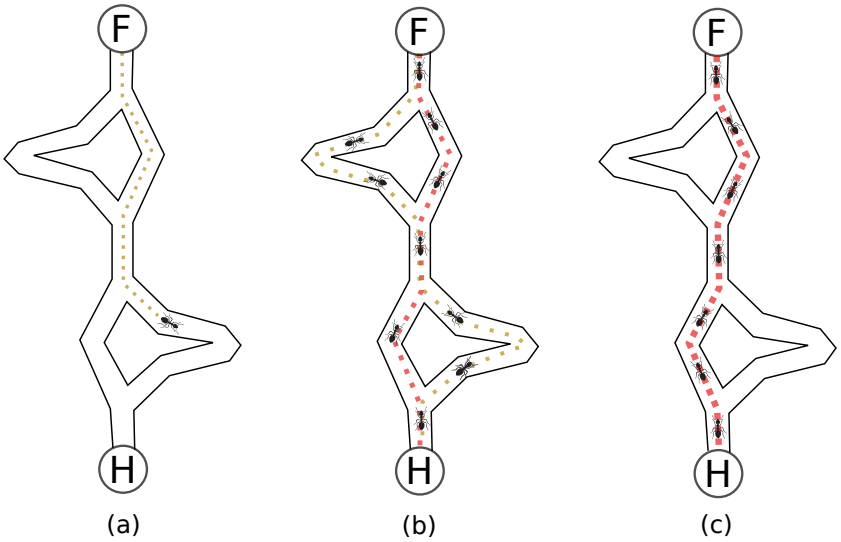


Abbildung 8: Entstehung des kürzesten Pfades [23]

Informationen, welche über Stigmergie ausgetauscht werden, sind lokal, d. h., sie können nur von Ameisen empfangen werden, die sich in direkter Nähe befinden. Andererseits müssen Ameisen nicht zu derselben Zeit an demselben Ort sein, um mit anderen Ameisen zu kommunizieren, sondern können zeitversetzt Informationen austauschen.

Der Prozess der Bildung des kürzesten Pfades wird in der Abbildung 8 dargestellt. Die Ameisen, welche die Futterstelle (F) entdeckt hatten und den Weg zum Ameisenhaufen (H) einschlagen, wählen auf einer Weggabelung mit gleicher Wahrscheinlichkeit den linken oder den rechten Pfad (a). Die Ameisen, die den kürzeren Pfad gewählt hatten, erreichen den Ameisenhaufen als erste, sodass sie als erste wieder den Weg zur Futterstelle aufnehmen. Aus diesem Grund steigt die Pheromonmenge auf dem kürzeren Pfad schneller (b). Ein Pfad mit höherer Konzentration an Pheromonen erscheint für andere Ameisen attraktiver. Auf lange Sicht wird der kürzeste Pfad verstärkt, die anderen Pfade werden dagegen durch die Verdampfung der Pheromone abgeschwächt. Durch diesen Prozess werden schließlich alle Ameisen auf den kürzeren Pfad gelenkt. Nach einer bestimmten Zeit wählen sämtliche Ameisen den kürzesten Pfad (c). In diesem Prozess wird eine höhere Leistung durch das Zusammenspiel vieler Ameisen erbracht, die jeweils nur einen Teil zur Gesamtlösung beitragen. Pheromonpfade fungieren als Langzeitspeicher des Suchprozesses, d. h., sie speichern alle Entscheidungen der Ameisen während der Suche in Form einer Pheromonlevel-Matrix ab. Während eine Ameise ihren Pfad konstruiert, wertet sie bereits von anderen Ameisen besuchte Pfade aus und entscheidet, welchen Pfad sie selbst nimmt. Das bedeutet wiederum, dass sie dadurch eine Entscheidung trifft, auf welchem Pfad sie ihre eigenen Pheromone hinterlässt. Die aktuelle Wahl der Ameise beeinflusst die Entscheidungen aller zukünftigen Ameisen.

Ein positives Feedback im Algorithmus entsteht dadurch, dass Ameisen Pfade mit stärkerem Pheromonlevel bevorzugen und immer dazu tendieren, diesen Pfaden zu folgen. Das hat jedoch den Nachteil, dass der Algorithmus zu schnell konvergiert und keine optimalen Resultate erreicht. Um diesem Prozess entgegenzuwirken, wird auch ein nega-

tives Feedback im System umgesetzt. Die Verdampfung der Pheromonpfade reduziert die Pheromonkonzentration an den Kanten des Graphen, sodass nur sehr gute Lösungen verstärkt werden und alle anderen mit der Zeit verschwinden. Die Verdampfung stellt eine Form des kollektiven Vergessens dar, was dem ganzen Algorithmus erlaubt, neue Bereiche mithilfe der Diversifizierung zu erschließen. Weiterhin begrenzt die Verdampfung das nach oben offene Steigen der Pheromonkonzentration. Um das negative Feedback im Algorithmus zu steuern, wird ein Verdampfungsfaktor benutzt. Der Faktor der Verdampfung der Pheromone darf nicht zu hoch oder zu niedrig angesetzt sein. Ist der Faktor zu hoch, wird der Effekt der kollektiven Suche unterbrochen, da Pfade schneller verdampfen, als sie von den Ameisen verstärkt werden können. Informationen über bereits gefundene Lösungen gehen dabei verloren. Ist der Faktor zu gering, wird der Prozess der Auslöschung der schlechten und guten, aber nicht sehr guten Lösungen nicht mehr ausreichend durchgeführt, sodass Ameisen überall Pfade mit hohen Pheromonwerten vorfinden. Aus diesem Grund stellt die Verdampfung der Pheromone eines der wichtigsten Elemente des Algorithmus dar. Im Algorithmus geht es nicht nur darum, den besten Pfad mehr als alle andere Pfade zu verstärken, sondern auch darum, alle ungünstigen Pfade soweit abzuschwächen, dass sie für Ameisen irrelevant werden [25].

Die Intensivierung und Diversifizierung werden in ACO mithilfe der α- und β-Parameter gesteuert. Je höher der Wert α, desto mehr wird die Intensivierung verstärkt, weil die Pheromonpfade dadurch für die Ameisen an Bedeutung gewinnen. Folglich, je kleiner α, desto stärker wird die Diversifizierung unterstützt, weil Ameisen den Pheromonen weniger Beachtung schenken. Parameter β funktioniert auf dieselbe Weise für die Distanz zwischen den Knoten des Graphen. Dabei stellt die Distanz zwischen den Knoten x und y bei einem NRP-Problem den Gewinn des Kunden y dar. Folglich müssen die beiden Parameter gleichzeitig verändert werden, um eine bessere Kontrolle über die Intensivierung und Diversifizierung zu erhalten. Im Folgenden werden Charakteristiken der Ameisenkolonie aufgelistet, die sie von anderen Algorithmen unterscheiden [25]:

- Die innere Parallelität ermöglicht es den Ameisen, den Suchraum parallel zu durchsuchen.
- Die Robustheit stellt sicher, dass die Kolonie in ihrer Aktivität fortfährt, wenn Individuen der Kolonie ausfallen.
- Die Dezentralisierung gewährleistet, dass die Kolonie keiner zentralen Autorität untergeordnet ist.
- Die Selbstorganisation garantiert, dass die Kolonie eine Lösung findet, welche nicht im Voraus bekannt ist.

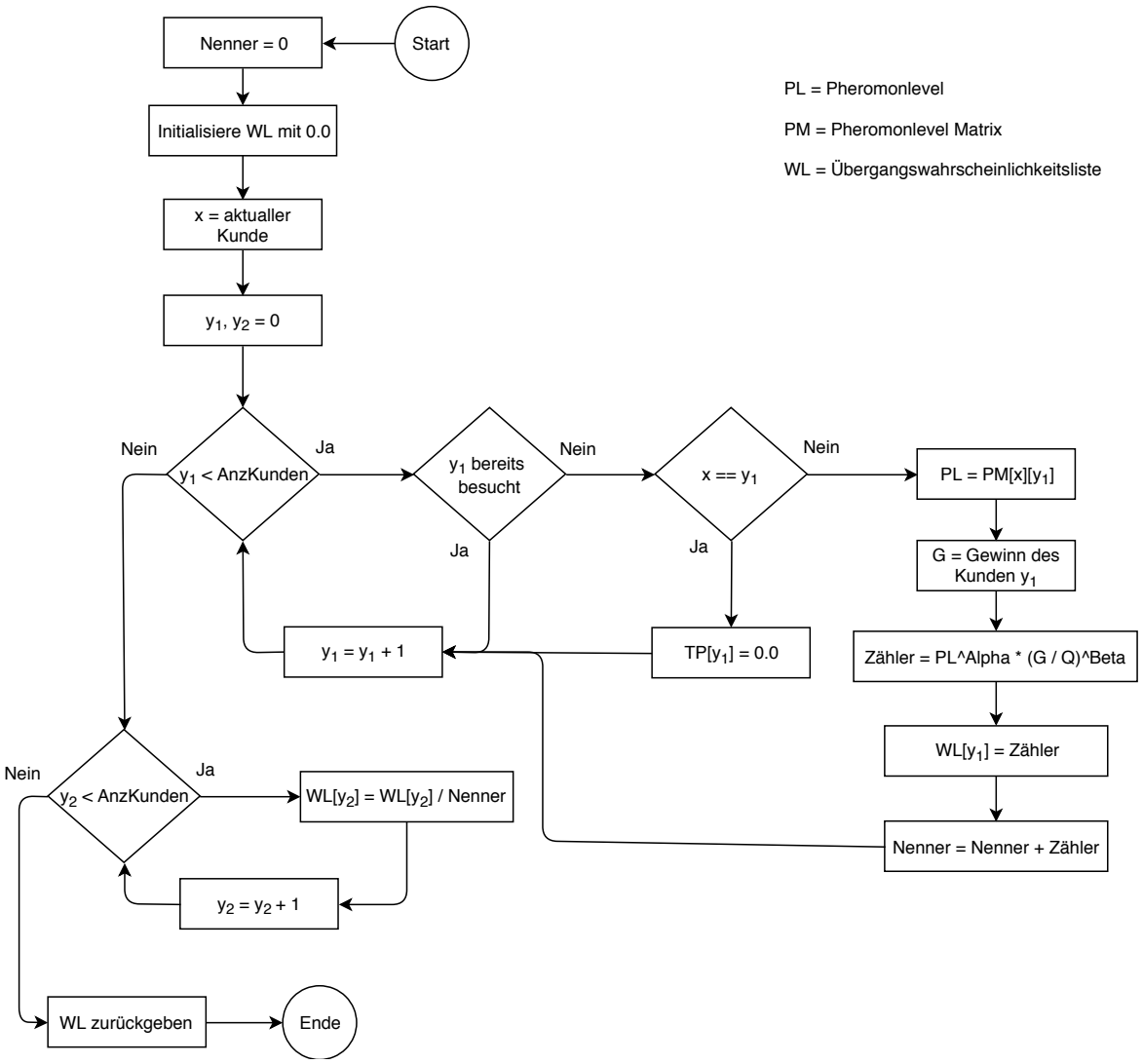


Abbildung 9: ACO: Berechnung der Übergangswahrscheinlichkeiten

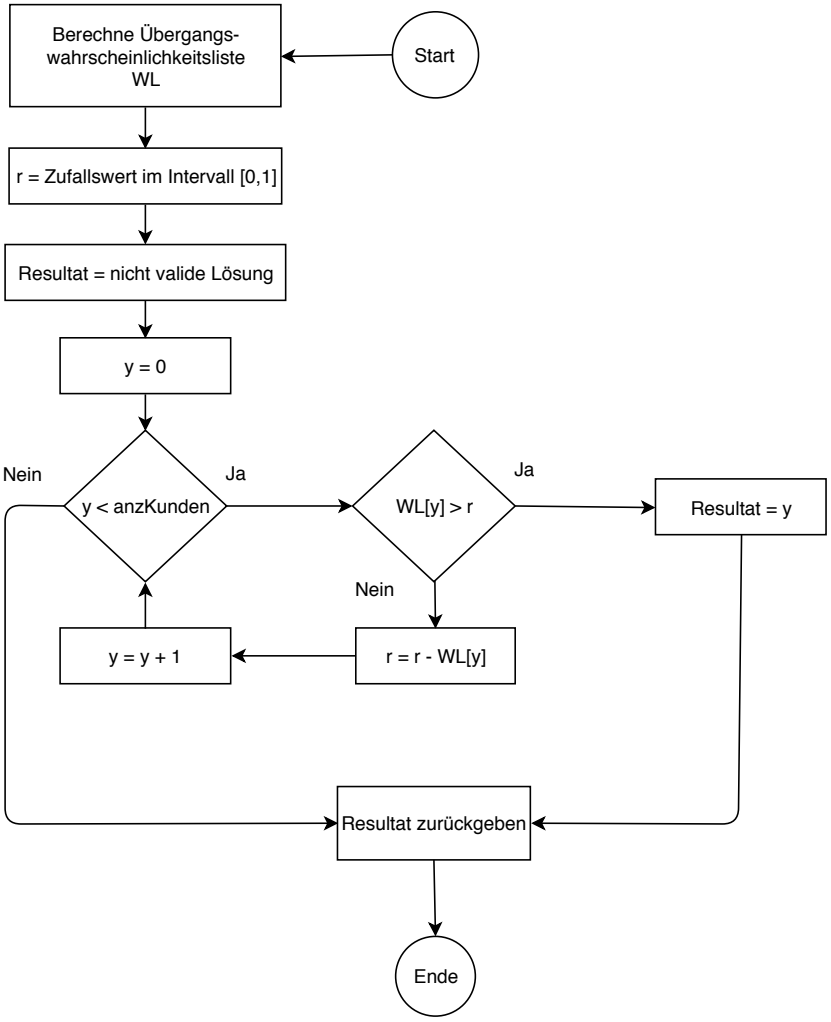


Abbildung 10: ACO: Bestimmung des nächsten Kunden

4.2.1 Programmablauf

Die ACO-Metaheuristik wird im Folgenden anhand drei Programmablaufpläne beschrieben, welche in den Abbildungen 9 – 11 zu sehen sind. Dabei stellt die Abbildung 9 den Prozess der Berechnung der Übergangswahrscheinlichkeiten für eine Ameise im aktuellen Knoten dar. In Abbildung 10 geht es um die Bestimmung des nächsten Knotens bzw. Kunden anhand der Übergangswahrscheinlichkeiten. Die Abbildung 11 präsentiert den gesamten ACO-Algorithmus. Im Folgenden wird auf einzelne Programmablaufpläne eingegangen, sodass die Funktionsweise des Algorithmus dadurch deutlich wird.

Die Berechnung der Übergangswahrscheinlichkeitsliste, wie sie in Abbildung 9 beschrieben wird, erfolgt anhand der klassischen Formel, welche bei Dorigo [23] angeführt ist. Dabei werden folgende Schritte unternommen:

1. Im ersten Schritt wird jedes Element der Übergangsliste mit 0 initialisiert. Dabei befindet sich die Ameise in einem bestimmten Knoten x. Gesucht sind Übergangswahrscheinlichkeiten zu allen anderen Knoten, wobei die Wahrscheinlichkeit, den aktuellen Knoten oder einen der bereits besuchten Knoten wiederzuwählen gleich 0 ist.

2. Im zweiten Schritt wird der Zähler der Formel berechnet: (Pheromonlevel zwischen den Knoten x und y) $^{\alpha}\cdot$ (Gewinn des Kunden y / Q) $^{\beta}$. Die temporäre Übergangswahrscheinlichkeit für Knoten y wird mit dem berechneten Zähler initialisiert. Der Nenner bildet die Summe aller Übergangswahrscheinlichkeiten, sodass er für jeden weiteren Knoten aufsummiert wird.

3. Im dritten Schritt, wenn die Liste der temporären Übergangswahrscheinlichkeiten für alle Knoten berechnet wurde, wird jeder Zähler durch den Nenner geteilt. Daraus entsteht die reale Übergangswahrscheinlichkeitsliste für alle Knoten. Die Summe der Wahrscheinlichkeiten ergibt immer 1. Damit wird ausgeschlossen, dass Ameisen in einem Knoten stecken bleiben, unschlüssig, welchen weiteren Knoten sie nehmen sollen, falls noch unbesuchte Knoten im Graphen vorhanden sind.

Anhand der berechneten Übergangswahrscheinlichkeiten wird im weiteren Schritt der nächste Knoten bzw. Kunde ausgewählt. Der PAP in der Abbildung 10 beschreibt den Prozess dieser Auswahl. Um den nächsten Knoten zu bestimmen, zu welchem die Ameise wechselt, werden folgende Schritte unternommen:

1. Im ersten Schritt wird die Übergangswahrscheinlichkeitsliste aus der Abbildung 9 berechnet, ein Zufallswert r im Bereich zwischen 0 und 1 gebildet sowie die Nummer des nächsten Knotens y = 0 initialisiert.

2. Im zweiten Schritt wird überprüft, ob die Übergangswahrscheinlichkeit für Knoten y größer r ist. Falls das der Fall ist, so wechselt die Ameise zum Knoten y.

3. Falls jedoch die Übergangswahrscheinlichkeit zu Knoten y kleiner r ist, so wird die Übergangswahrscheinlichkeit zu Knoten y von r abgezogen. Man erhöht die

Nummer des Knotens mit y = y + 1 und startet mit (2) noch einmal. Da die Summe der Wahrscheinlichkeiten immer 1 ergibt, wird damit ausgeschlossen, dass kein weiterer Knoten gewählt wird. Die Übergangswahrscheinlichkeit für den letzten Knoten beträgt stets 100 %.

Die beiden Abbildungen 9 und 10 waren als Module für den ACO-Algorithmus notwendig. In der Abbildung 11 wird der ganze Algorithmus beschrieben, indem er auf die bereits eingeführten Module zurückgreift.

Die Kernidee des Algorithmus besteht darin, dass, ausgehend von dem aktuellen Knoten, immer neue unbesuchte Knoten zum Pfad mithilfe der Übergangswahrscheinlichkeiten hinzugenommen werden. Gleichzeitig wird bei der Auswahl des nächsten Knotens der Pheromonlevel um den Verdampfungsfaktor verringert sowie um die Menge der Pheromone, welche die aktuelle Ameise hinterlässt, verstärkt. Bei der Verstärkung spielt die Distanz zum nächsten Knoten bzw. der erwartete Gewinn des Kunden eine wichtige Rolle. Je höher die Distanz bzw. der Gewinn, desto mehr Pheromone werden von der Ameise hinterlassen. Der Algorithmus bricht ab, wenn alle Knoten besucht wurden oder wenn ein nicht valider Pfad konstruiert wurde. Die einzelnen Phasen werden im Folgenden im Detail erläutert:

1. Im ersten Schritt wird die Initialisierungsphase durchgeführt. Dabei werden folgende Parameter initialisiert: Anzahl der Ameisen $A_{max}, \alpha, \beta, \rho$ und Q. Die Liste der besuchten Knoten ist anfangs leer. Die Pheromonlevelmatrix PM wird mit den Zufallswerten im Bereich zwischen 0 und 1 initialisiert.

2. Für jede Ameise wird folgende Berechnung des Pfades durchgeführt. Man wähle einen zufälligen Knoten bzw. Kunden x. Ausgehend von x wird der nächste Knoten y, wie in der Abbildung 10 gezeigt, bestimmt. Eine temporäre Kopie des aktuell validen Pfades wird erstellt.

3. Der aktuelle Pfad wird um den Knoten x erweitert und evaluiert. Falls der Pfad dadurch nicht valide wird, so wird die valide temporäre Kopie zurückgegeben.

4. Ist der entstandene Pfad valide, so wird der Pheromonlevel zwischen den Knoten x und y mit der Formel: $(1 - \rho)\cdot$ der alte Pheromonlevel + Gewinn des Kunden y/Q aktualisiert. x wird zu der Liste der besuchten Knoten hinzugefügt und y wird zum neuen aktuellen Knoten $x = y$.

5. Der Prozess wird wiederholt (2), solange im Graphen noch unbesuchte Knoten existieren oder eine nicht valide Lösung produziert wurde.

6. Der neu erstellte Pfad wird mit dem besten bis dahin gefundenen Pfad verglichen, und falls der neue Pfad besser ist, d. h. höheren Gewinn erbringt, wird er zum neuen besten Pfad.

Im Pareto-Fall wird jeder berechnete Pfad einem Dominanz-Archiv [40] hinzugefügt, wo alle dominierten Lösungen aussortiert werden, sodass am Ende eine Pareto-optimale Lösungsmenge übrigbleibt.

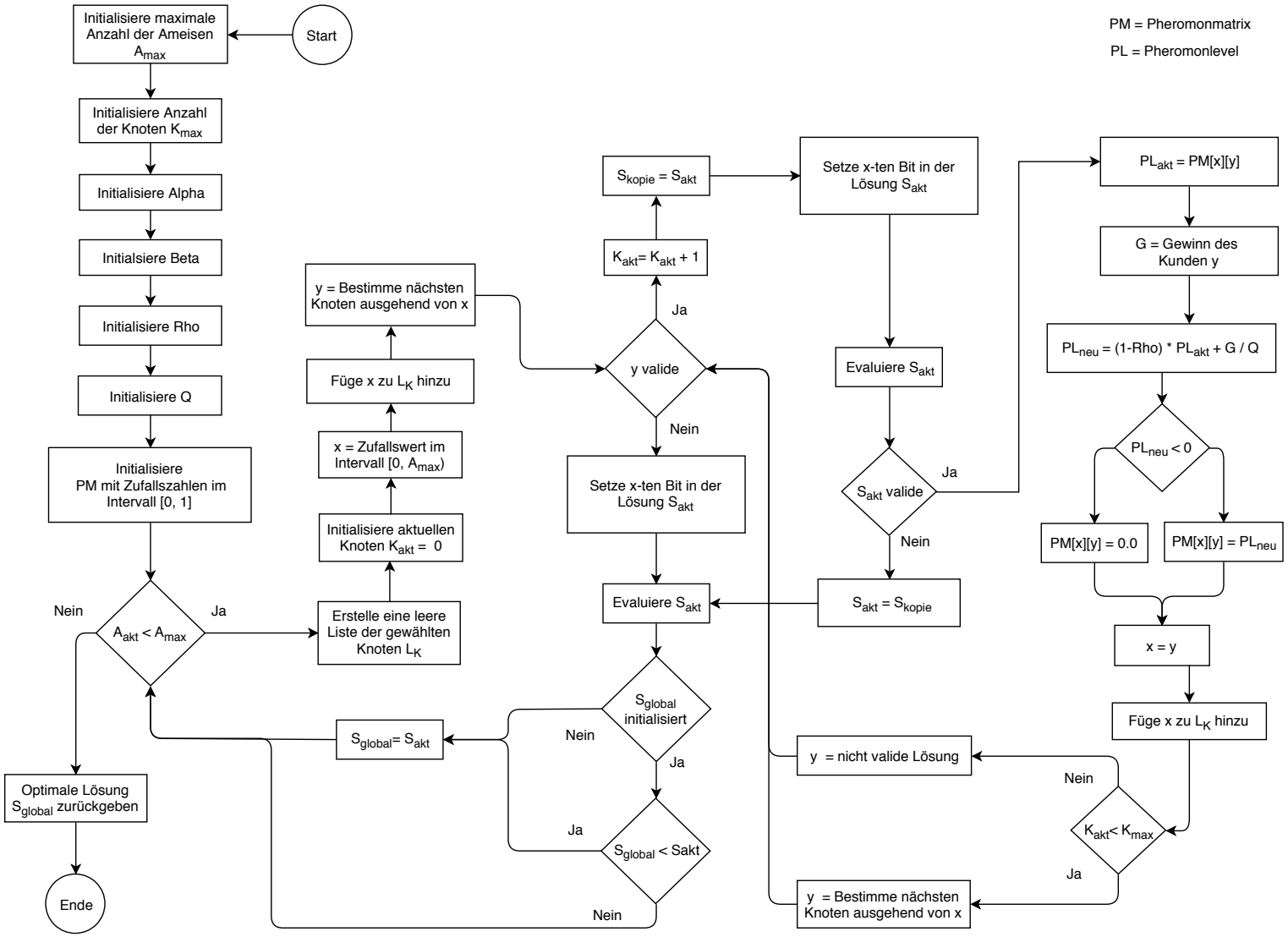


Abbildung 11: Nicht-Pareto Ant Colony Optimization PAP

4.3 Random Search

Random Search steht für eine zufällige Suche und gehört zu den globalen stochastischen Optimierungsalgorithmen. Jede neue Lösung ist in RS von der vorhergehenden unabhängig. Random Search ist minimal und braucht nur einen Lösungskandidaten und eine Fitnessfunktion zur Auswertung der Lösungen, um zu funktionieren. Die schlechteste Leistung von Random Search kann schlechter sein als die Dimension des gesamten Suchraumes, da Random Search keinen Speicher besitzt und Lösungen immer blind geraten werden. Random Search ist eine extrem einfache Methode. Sie erforscht den Suchraum durch die zufällige Auswahl der Lösungen und wertet sie mithilfe der Zielfunktion aus. Bei RS handelt es sich um eine sehr primitive Strategie, welche selten genutzt wird. Nichtsdestotrotz ist diese Methode manchmal von Bedeutung, z. B. für einen Vergleich mit anderen Algorithmen, indem man die Algorithmen die gleiche Zeit oder gleiche Anzahl von Iterationen laufen lässt, um sicherzustellen, dass diese Algorithmen eine bessere Leistung als Random Search erbringen. Für diesen Zweck wird RS auch in dieser Arbeit verwendet und bildet die untere Schranke der Leistung für andere Algorithmen.

Der Programmablaufplan von Random Search, der in der Abbildung 12 zu sehen ist, wird wie folgt beschrieben:

1. Im ersten Schritt werden alle benötigten Parameter: aktuelle Iteration i sowie die maximale Anzahl der Iterationen I_{max} initialisiert.

2. Im zweiten Schritt wird eine Lösung aus der Nachbarschaft der aktuellen Lösung S_{akt} erstellt und evaluiert. Der Algorithmus wechselt immer zu der neuen Lösung.

3. Im dritten Schritt wird verglichen, ob die neue Lösung die beste bis dahin gefundene Lösung an Fitness übersteigt. Falls das der Fall ist, wird die neue Lösung in S_{best} gespeichert.

4. Im Schritt vier wird überprüft, ob die maximale Anzahl der Iterationen bereits erreicht ist. Falls das der Fall ist, wird der Algorithmus verlassen und die beste Lösung zurückgegeben, sonst wird die Schleife (2) wiederholt.

Im Pareto-Fall werden alle Lösungen ins das Dominanz-Archiv [40] hinzugefügt, sodass dominierte Lösungen aussortiert werden.

4.3.1 Implementierungsdetails

In der Umsetzungsphase wurden mehrere Operatoren zur Nachbarfindung ausprobiert. Im ersten Entwicklungszyklus wurde ein Flip-Mutations-Operator benutzt, wie er bei GA beschrieben ist, der eine Lösung der Länge k in einer Schleife durchlief und in jeder Iteration zufallsbasiert Bits kippte. Dieser Mutationsoperator hat sich als sehr schlecht erwiesen, da dadurch Lösungen aus dem ganzen Suchraum zufällig generiert wurden. Da aber Lösungen aller NRP-Probleme durch einen festgelegten Kostenfaktor begrenzt sind, lieferte dieser Operator in den meisten Fällen nicht valide Lösungen. Insbesondere

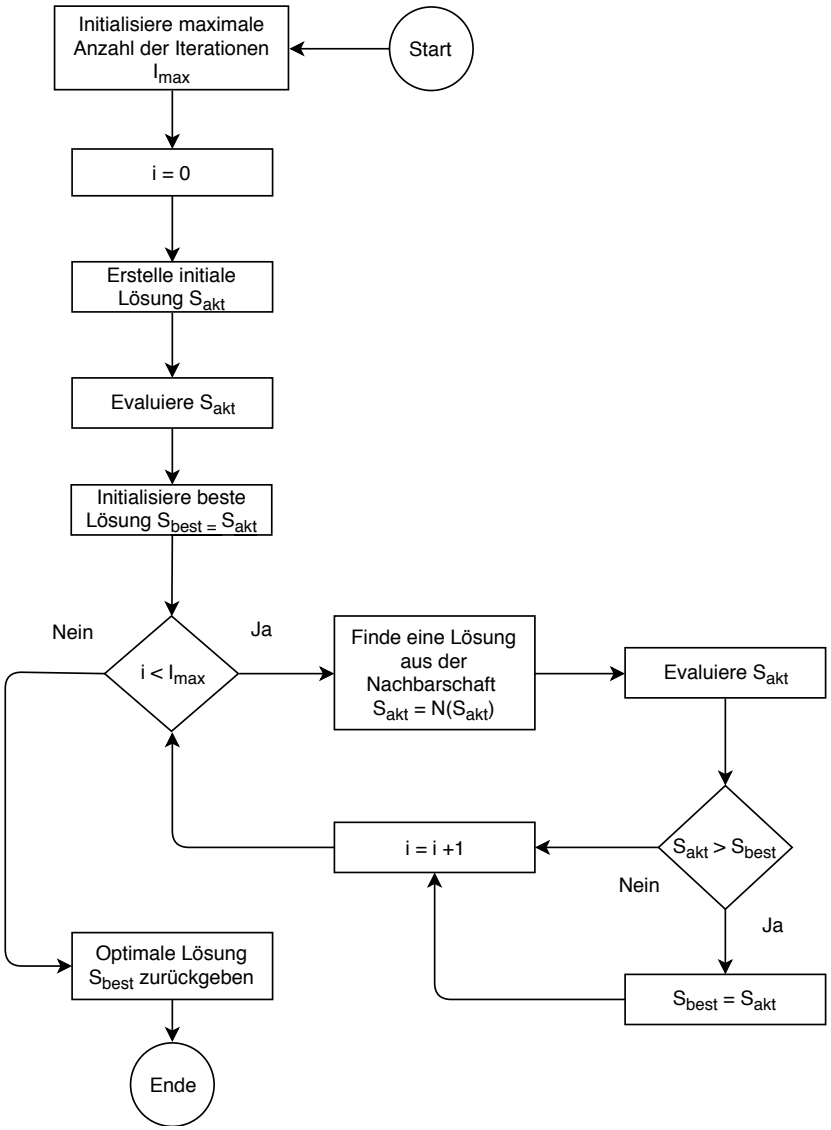


Abbildung 12: Nicht-Pareto Random Search PAP

war das beim Kostenfaktor 0.3 bemerkbar, sodass manchmal nur ein Dutzend Lösungen pro Durchlauf mit mehreren Tausend Iterationen entstanden. Aus diesem Grund wurde FlipOrExchange-Operator gewählt, um die Kosteneinschränkung des Datensatzes bei der Nachbarsuche adäquat zu verarbeiten. Dadurch wurde in jeder Iteration nur ein Bit ausgetauscht oder gekippt, d. h., es wird immer zu einem direkten Nachbarn gewechselt, aber im Vergleich z. B. zu Hill Climbing ist die Fitness der Nachbarlösung für den Wechsel irrelevant. Das brachte ein viel besseres Verhalten des Algorithmus zustande, sodass in den meisten Fällen valide Lösungen generiert wurden, bis die Grenze zu nicht validen Lösungen überschritten wurde. Und da RS zum zufälligen Nachbarn wechselt, ohne auf seinen Fitnesswert zu achten, verlor sich RS anschließend in Suchbereichen mit nicht validen Lösungen und kam nur selten wieder heraus.

Außerdem wurde anfangs mit einer zufälligen initialen Lösung gearbeitet. Allerdings, wiederum bedingt durch den Kostenfaktor des Datensatzes, wurde oft eine nicht valide initiale Lösung generiert. Das führte dazu, dass der Algorithmus ein sehr schlechtes Verhalten aufwies, da er eine Zeit brauchte, um den nicht validen Bereich des Suchraumes, wo er gestartet hatte, wieder zu verlassen. Aus diesem Grund wurde im weiteren Entwicklungszyklus eine andere Strategie zur Initialisierung gewählt. Alle Bits der initialen Lösung wurden auf 0 gesetzt. Das hat ein besseres Verhalten zur Folge, da für jeden Kostenfaktor garantiert ist, dass eine Lösung mit dem Fitnesswert 0 immer eine valide Lösung darstellt.

4.4 Hill Climbing

Das Prinzip des traditionellen Hill-Climbing-Algorithmus [25] lautet wie folgt: Man startet mit einer initialen Lösung, dann wird ein Nachbar ausgewählt und die Fitnesswerte beider Lösungen werden verglichen. Wenn die Modifikation zur Erhöhung des Fitnesswertes geführt hat, wird sie akzeptiert. Die neue Lösung wird zum Ausgangspunkt für eine weitere Iteration genutzt. Ist es nicht der Fall und die Modifikation führt zu einer Verringerung des Fitnesswertes, so wird die Anfangslösung beibehalten und mit einer weiteren Iteration fortgefahren. Dieser Prozess wird iterativ durchgeführt, bis eine Abbruchbedingung erreicht ist. Die Vorteile dieser Methode liegen in der Einfachheit der Implementierung sowie ihrer Schnelligkeit, da die Operatoren sehr einfach und knapp gehalten sind. Bei Hill Climbing handelt es sich um eine Heuristik, da der Algorithmus keine Möglichkeit hat, das lokale Optimum zu verlassen. Diese Heuristik konvergiert gegen das globale Optimum, wenn die Zielfunktion eines Problems ein einziges Optimum besitzt. Hat die Zielfunktion dagegen mehrere Optima, wird diese Methode in dem ersten lokalen Optimum stecken bleiben. In diesem Fall kann es in der Heuristik keinen weiteren Fortschritt mehr geben.

In der Abbildung 13 wird der Programmablaufplan für Hill-Climbing der Vollständigkeit halber angegeben, obwohl er in dieser Arbeit nicht direkt umgesetzt wurde. Er spielt aber im nächsten Abschnitt bei der SA eine Rolle, deswegen wird im Folgenden seine Funktionsweise anhand des Programmablaufplans erläutert:

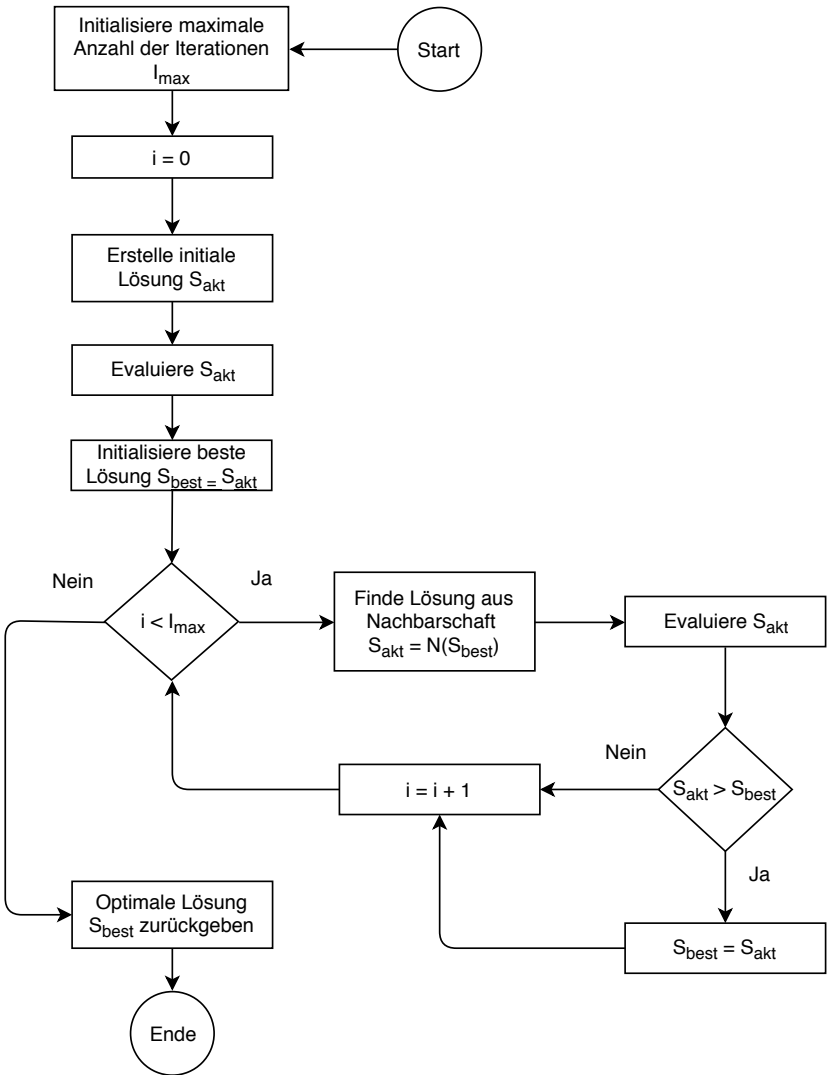


Abbildung 13: Nicht-Pareto Hill Climbing PAP

1. Im ersten Schritt erfolgt die Initialisierung der Variablen. Eine initiale Lösung wird erstellt. Die beste Lösung S_{best} wird mit der initialen Lösung initialisiert.

2. Im zweiten Schritt wird aus der besten Lösung S_{best} eine neue Lösung aus der Nachbarschaft generiert.

3. Im dritten Schritt wird überprüft, ob die neue Lösung eine bessere Fitness aufweist als die beste Lösung. Falls das der Fall ist, so erfolgt der Wechsel, sodass die neue Lösung zur besten Lösung wird.

4. Im vierten Schritt wird überprüft, ob die maximale Anzahl an Iterationen überschritten wurde. Falls das der Fall ist, wird der Algorithmus verlassen, ansonsten wird eine weitere Iteration (2) ausgeführt.

Man könnte auf die Idee kommen, den Algorithmus zu verbessern, indem man ihn parallel aus verschiedenen Startpunkten laufen lässt, sodass am Ende die beste gefundene Lösung gewählt wird. Dieses Vorgehen hat jedoch keine besseren Aussichten auf Erfolg als die Originalmethode. Das hat folgenden Hintergrund: Besitzt das Problem eine hohe Anzahl an lokalen Optima, so führt die Ausführung aus verschiedenen Startpunkten dazu, dass jede Ausführung mit einer sehr hohen Wahrscheinlichkeit in einem lokalen Optimum landet. Die Ausführung aus verschiedenen Startpunkten führt ausschließlich zur Erhöhung der verwendeten Ressourcen, bringt aber keine weiteren Vorteile mit sich. Offensichtlich ist diese Vorgehensweise ineffektiv und es werden andere Techniken, wie z. B. Simulated Annealing, benötigt, die dabei helfen, lokale Optima zu verlassen.

4.5 Simulated Annealing

Simulated Annealing [26, 27] ist eine Metaheuristik, welche auf Hill Climbing sowie Random Search basiert. Ihre Funktionsweise ähnelt Hill Climbing mit der Nebenbedingung, dass in Abhängigkeit von der Temperatur eine Wahrscheinlichkeit besteht, schlechtere Lösungen zu akzeptieren, um das Anhalten im lokalen Optimum zu vermeiden. Dadurch kann SA ein lokales Optimum verlassen. Es werden ungünstige Zwischenlösungen akzeptiert, um ein globales Optimum zu finden. Ist die Temperatur hoch, so ist die Wahrscheinlichkeit, eine schlechtere Lösung zu akzeptieren, hoch, was das Verhalten von Random Search darstellt. Sinkt die Temperatur, so wird die Wahrscheinlichkeit, schlechtere Lösungen zu akzeptieren, geringer, sodass der Algorithmus die Funktionsweise von Hill Climbing nachahmt. Um zu entscheiden, ob eine Lösung akzeptiert wird, wird folgende Formel zur Berechnung der Akzeptanzwahrscheinlichkeit genutzt: $p = e^{-\frac{\Delta S}{T}}$, wobei ΔS die Differenz der aktuellen und der neuen Lösung darstellt. Das Zufallsprinzip bei der Wahl der nächsten Lösung wurde eingeführt, um die Wahrscheinlichkeit zu verringern, in einem lokalen Optimum anzuhalten. Bei hohen Temperaturen liegt p nahe bei 1, deswegen wird die Mehrheit der Lösungen beliebiger Qualität akzeptiert. Bei kleinen Temperaturen strebt p gegen 0, sodass die Mehrheit der Lösungen, welche zur Verschlechterung führen, abgelehnt werden. Außerdem hängt die Wahrscheinlichkeit, eine schlechtere Lösung zu akzeptieren, von ΔS ab. Wenn die neue Lösung eine schlechtere

Fitness aufweist, dann wird sie mit der Wahrscheinlichkeit akzeptiert, welche sinkt, wenn die Differenz der Fitnesswerte der aktuellen und der neuen Lösung steigt. Es gibt aber eine Bedingung, die für alle Phasen des Algorithmus gilt, dass ungeachtet der Temperatur eine Lösung mit einem besseren Fitnesswert immer akzeptiert wird. Für gewöhnlich startet Simulated Annealing mit einer hohen Temperatur, welche kontinuierlich sinkt, indem die aktuelle Temperatur T_{akt} mit einem Faktor $\alpha < 1$ multipliziert wird. Grundsätzlich gilt, je langsamer das Abkühlen, desto besser ist es zum Finden optimaler Lösungen. Das hat folgenden Hintergrund: Bei hohen Temperaturen bietet SA eine hohe Diversifizierung, aber beinahe keine Intensivierung. Erfolgt das Abkühlen zu schnell, dann hat der Algorithmus keine Möglichkeit, für eine ausreichende Diversifizierung zu sorgen. Und da Hill Climbing eine hohe Intensivierung, aber beinahe keine Diversifizierung bietet, sollte die Intensivierung in einem vielversprechenden Bereich erfolgen, der zuvor durch die Diversifizierung gefunden wurde. Aus dieser Sicht ist es besser, wenn die Phasen der Random Search sowie Hill Climbing nacheinander erfolgen, sodass Diversifizierung und Intensivierung abwechselnd in unterschiedlichem Maße stattfinden, was wiederum nur im mittleren Temperaturbereich möglich ist. Diese Tatsache hat praktische Konsequenzen für die Implementierung. Um das Abkühlen im Algorithmus zu verlängern, sollte nicht die Anfangstemperatur höher gesetzt werden, da in diesem Fall nur die Phase vom Random Search länger wird, sondern es sollte stattdessen die Kühlrate α kleiner gesetzt werden.

In der Abbildung 14 ist zu sehen, dass der SA-Programmablaufplan ähnlich zu PAP von Hill Climbing sowie Random Search aussieht. Der genaue Ablauf wird in folgender Abbildung erklärt:

1. Die benötigten Variablen: aktuelle Temperatur T_{akt}, Endtemperatur T_{end}, Kühlrate α werden initialisiert.

2. Eine Lösung S_{tmp} wird aus der Nachbarschaft von S_{akt} generiert.

3. Ist die neue Lösung S_{tmp} besser als S_{akt} so wird sie sofort akzeptiert, man verringert die Temperatur und man fährt mit (2) fort.

4. Ist die Lösung S_{tmp} schlechter, hat sie dennoch eine Wahrscheinlichkeit $p = e^{-\frac{\Delta S}{T_{akt}}}$ akzeptiert zu werden. Dabei gilt, je höher T_{akt}, desto höher ist die Wahrscheinlichkeit.

5. Anschließend wird die aktuelle Temperatur exponentiell verringert $T_{akt} = \alpha \cdot T_{akt}$.

6. Danach werden alle Schritte ab (2) wiederholt, bis die Temperatur ihren minimalen Level erreicht und der Algorithmus anhält.

Simulated Annealing nutzt überhaupt keinen Speicher und ist somit nicht in der Lage Rückschlüsse aus der Vergangenheit zu ziehen. Im nächsten Kapitel soll ein weiterer lokaler Algorithmus betrachtet werden, welcher eine gegensätzliche Strategie zu SA nimmt und einen adaptiven Speicher bei der Suche benutzt.

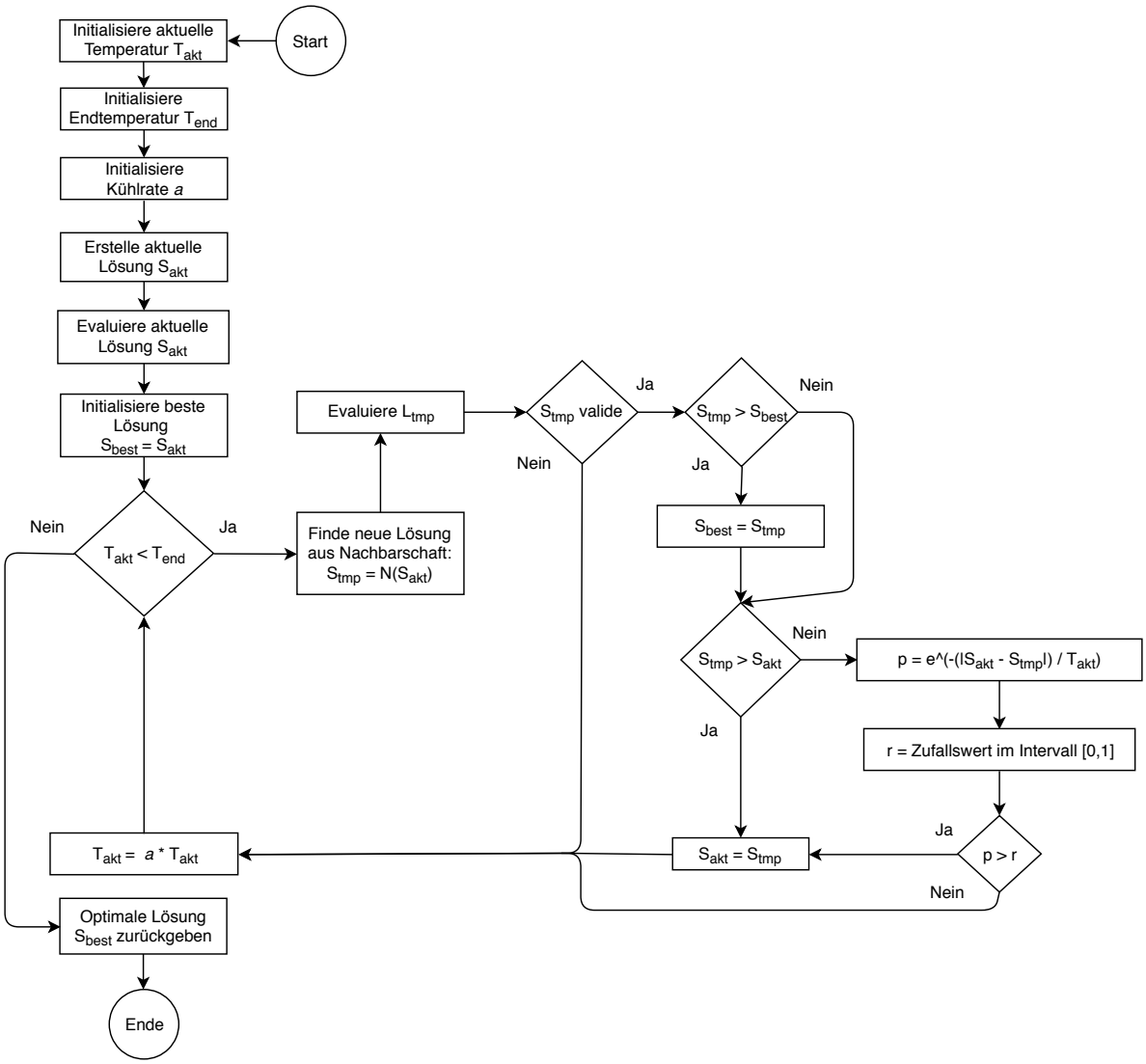


Abbildung 14: Nicht-Pareto Simulated Annealing PAP

4.6 Tabu Search

Tabu Search ist ein lokales metaheuristisches Verfahren zum Lösen der kombinatorischen Probleme. Diese Methode wurde von F. Glover [28] erfunden, um die Möglichkeit zu erhalten, lokale Optima der Zielfunktion zu verlassen. TS nutzt eine lokale Suche, um den Suchraum nach optimalen Lösungen zu erforschen. Anders als bei evolutionären Algorithmen wird bei der klassischen Tabu-Suche in jeder Iteration von nur einer Lösung ausgegangen. Die iterative Vorgehensweise erlaubt der Tabu Search, sofort angehalten zu werden, wenn die gewünschte Qualität der Lösungen erreicht ist. Im Gegensatz dazu konvergiert Simulated Annealing erst bei niedrigen Temperaturen gegen ein Optimum, was das Anhalten des Algorithmus zu beliebiger Zeit problematisch macht.

TS versucht den Suchraum eines Problems intelligent zu erforschen, wozu ein adaptiver Speicher gewählt wird. Der adaptive Speicher in Form einer Tabu-Liste ist ein Mechanismus, welcher von Tabu Search benutzt wird, um gute Lösungen bereits in früheren Stadien der Suche zu entdecken. Die Funktionsweise von TS gründet auf zwei Überlegungen. Erstens, können Optimierungsprobleme gelöst werden, indem man den besten Zug macht, welcher aktuell zur Verfügung steht. Zweitens, widmet sich die Tabu Search, statt Zeit darauf zu verschwenden, weniger attraktive Bereiche nach globalen Optima zu überprüfen, der Untersuchung der Bereiche, bei denen bereits bekannt ist, dass sie Lösungen von guter Qualität enthalten [28]. Außerdem gilt für alle lokale Suchalgorithmen die Annahme, dass eine einfache strategische Suche im Schnitt bessere Lösungen produzieren kann als Random Search.

TS startet mit einer initialen Lösung, erstellt eine Liste der Nachbarn und wechselt dann zu dem Nachbarn mit der besten Fitness. Die Suche der nachfolgenden Lösung ist ausschließlich auf die Nachbarn der aktuellen Lösung beschränkt, ausgenommen die Lösungen, welche sich gerade in der Tabu-Liste befinden. Die Tabu.Liste beinhaltet Lösungen, die als Tabu klassifiziert wurden und bei der Nachbarsuche nicht gewählt werden dürfen. Mithilfe der Tabu-Liste modelliert der Algorithmus eine rudimentäre Form des Kurzzeitgedächtnisses, das alle zuvor besuchten Lösungen speichert. Da aber die Tabu-Liste begrenzt ist und eine bestimmte Länge k besitzt, wird die Erfahrung in Form besuchter Lösungen nach k Schritten wieder vergessen. Somit ist die Tabu-Liste für die Diversifizierung im Algorithmus verantwortlich, indem sie Lösungen speichert, welche nicht wiedergewählt werden dürfen. Die Tabu-Liste wurde eingeführt, um das Problem der Zyklen aufzulösen. Da aber jedes Optimierungsproblem eine andere Anzahl und Anordnung an Optima besitzt, ist offensichtlich, dass die Größe der Tabu-Liste eine wichtige Rolle beim Verlassen der Zyklen spielt. Wird die Tabu-Liste als zu klein gewählt, so tendiert die Suche bei einer eingeschränkten Anzahl der Nachbarn dazu, immer wieder die gleichen Lösungen zu wählen. Wird die die Größe der Tabu-Liste erhöht, sinkt die Wahrscheinlichkeit, dieselben Lösungen mehrmals zu besuchen. Wird die Größe der Tabu-Liste zu hoch gesetzt, so wird auf ein anderes Problem gestoßen, und zwar, dass Bereiche übersprungen werden, welche möglicherweise gute Lösungen enthalten. Mithilfe der angemessenen Werte für die Größe der Tabu-Liste, sinkt die Wahrscheinlichkeit der Entstehung der Zyklen drastisch. Somit wird der Algorithmus durch die erlaubten Züge gesteuert und weniger durch die Zielfunktion [29].

In der Implementierung wurde eine statische Tabu-Liste, mittels der Ringspeicher-Datenstruktur umgesetzt, sodass jede neue Lösung, welche hinzugefügt wird, die älteste Lösung aus der Tabu-Liste verdrängt. Um den gleichzeitigen Vorteil einer kleinen und einer großen Tabu-Liste auszunutzen, kann es sinnvoll sein, die Größe der Tabu-Liste während des Suchprozesses zu verändern. Eine variable Tabu-Liste kann als eine Erweiterung des Algorithmus betrachtet werden, übersteigt aber den Umfang dieser Arbeit.

Bei der Nachbarsuche ist es oft nicht möglich, alle Nachbarn der aktuellen Lösung zu evaluieren, weil z. B. ein NP-hartes Problem vorliegt. Aus diesem Grund überprüft TS während der Ausführung nicht alle direkten Nachbarn, sondern nur eine Teilmenge davon. Diese Teilmenge reicht aber aus, um gute Entscheidungen zu treffen, welche die Suche zum globalen Optimum führen [11]. In der aktuellen Implementierung der TS wird die Anzahl der besuchten Nachbarn durch die Parameterwahl eingeschränkt. Die Einschränkung der Nachbarn hat aber die praktische Konsequenz, dass die Suche in Regionen geleitet wird, welche möglicherweise unbesucht bleiben würden, wenn alle möglichen Nachbarn zur Verfügung ständen. Die komplette Funktionsweise des Algorithmus wird mit dem folgenden Programmablaufplan beschrieben:

1. Im ersten Schritt werden Parameter wie Länge der Tabu-Liste, Anzahl der Nachbarn sowie die Anzahl der Iterationen festgelegt.

2. Im zweiten Schritt werden die Nachbarn der aktuellen Lösung erstellt und ausgewertet.

3. Im dritten Schritt werden daraus alle Lösungen entfernt, welche sich gerade in der Tabu-Liste befinden.

4. Anschließend wird im vierten Schritt die neue aktuelle Lösung gewählt.

5. Für den Nicht-Pareto-Fall wird die Lösung mit der besten bis dahin gefundenen Lösung verglichen, die zur besten Lösung wird, falls sie eine bessere Fitness aufweist.

6. Danach wird die aktuelle Lösung im Schritt sechs zu der Tabu-Liste hinzugefügt.

7. Anschließend wird die Schleife wiederholt (2), bis das Abbruchkriterium erfüllt ist.

In der Pareto-Implementierung werden alle Nachbarn außer den Lösungen in der Tabu-Liste in das Dominanz-Archiv [40] eingefügt, sodass sämtliche dominierten Lösungen aussortiert werden. Anschließend wird die letzte gefundene Pareto-optimale Lösung als neue aktuelle Lösung gewählt.

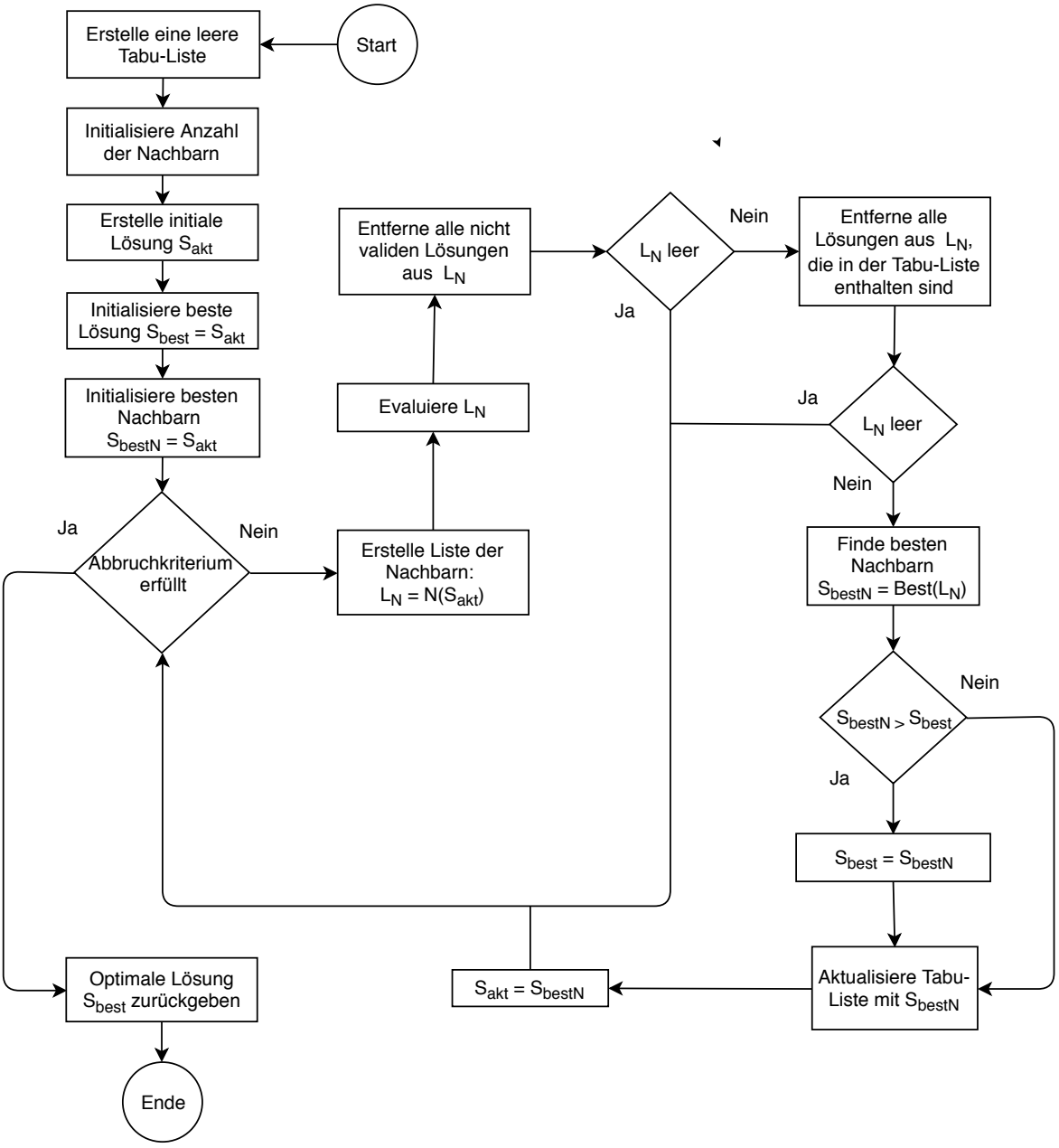


Abbildung 15: Nicht-Pareto Tabu Search PAP

4.7 Parameterwahl der Algorithmen

In den oberen Abschnitten, in denen es um die Beschreibung der Algorithmen ging, wurde das Thema der Parameterwahl nicht angesprochen, weil sie eine große Bedeutung für das Verhalten der Algorithmen hat und einen eigenen Themenabschnitt benötigt.

Die optimale Parameterwahl einer Metaheuristik, die theoretisch empfohlen werden könnte, ist in der Praxis nicht anwendbar, da Probleme und somit auch die Parameter zu spezifisch sind, um allgemeinen Richtlinien zu folgen. In der Praxis haben Anwender bei der Auswahl geeigneter Technik für einen konkreten Problembereich entweder die Möglichkeit, in der Literatur nach ähnlichen Einsatzfeldern und genutzten Methoden zu suchen, oder sie können mehrere unterschiedliche Techniken umsetzen und analysieren, um dann erfahrungsbasiert ihre eigene Meinung zu der jeweiligen Technik zu bilden. Die Beschreibung der Algorithmen in der Literatur behandelt oft nur die grundlegenden Konzepte und es finden sich kaum praktische Hinweise zur Gestaltung der Parameter. Aus diesem Grund soll an dieser Stelle der zweite Fall, nämlich die Erfahrung zur Parameterwahl, geschildert werden.

Tabelle 3 gibt einen Überblick über die genutzten Parameter anhand des nrp-e1-Datensatzes. Dieser Datensatz ist mittelgroß, sodass er repräsentativ für kleinere, aber auch für größere Datensätze die Parameterwahl beschreibt. Im Weiteren wird die Parameterwahl für den jeweiligen Algorithmus erläutert. Da aber die FlipOrExchange-Mutation für GA, TS, SA und RS benutzt wurde, so soll als Erstes auf die Parameterwahl der Mutation eingegangen werden, bevor dann für jeden Algorithmus spezifische Parametereinstellungen besprochen werden.

FlipOrExchange: Als Richtwert für die Parametereinstellung des FlipOrExchange-Operators hat sich die Pareto-Regel 20/80 etabliert. Das bedeutet, 20 % Wahrscheinlichkeit für die Verwendung des Flip-Operators sorgen für eine ausreichende Diversifizierung der Lösung und 80 % Wahrscheinlichkeit für Exchange-Operator für eine gute Intensivierung der untersuchten Bereiche. Hierbei kam es jedoch auch zu Abweichungen, bei denen eine starke Intensivierung nicht nötig war, wie es bei Nicht-Pareto-Problemen der Fall war. Bei Nicht-Pareto-Algorithmen wurde oft mit 50/50-Wahrscheinlichkeiten gearbeitet, um für ein schnelleres Konvergenzverhalten zu sorgen. Allerdings auch im Nicht-Pareto-Fall konnte auf den Exchchange-Operator bzw. die Intensivierung nicht komplett verzichtet werden, da die Lösungen sonst aufgrund des Kostenfaktors schnell die Grenze zu nicht validen Lösungen überschritten.

Genetic Algorithm Es ist in der ersten Zeile der Tabelle 3 zu sehen, dass für den Genetischen Algorithmus vier Parameter benutzt wurden: Populationsgröße 500, Kreuzungsrate 0.7, Mutationsrate 0.2 sowie 200.000 Iterationen. Eine umfangreiche Population bedeutet das gleichzeitige Verarbeiten vieler Lösungen und erhöht die Laufzeit einer Iteration. Da jedoch viele Lösungen aus dem Suchraum untersucht werden, ist die Wahrscheinlichkeit, gegen ein globales Optimum zu konvergieren, höher als wenn eine kleine Populationsgröße genommen wird [21]. Die Populationsgröße hat einen entscheidenden Einfluss auf die Intensivierung. Die Anzahl der Iterationen wurde problemspezifisch auf

Algorithmus	Parameter	Werte
Genetic Algorithm	Populationsgröße	500
	Kreuzungsrate	0.7
	Mutationsrate	0.2
	#Iterationen	200.000
Ant Colony	#Ameisen	500
	α	10
	β	10
	ρ	0.1
	Q	10500
Simulated Annealing	Starttemperatur	1000
	Endtemperatur	1
	Abkühlungsrate	0.0001
	Mutationsrate	0.5
Tabu Search	#Iterationen	1000
	#Nachbarn	100
	#Tabu Liste	100
	Mutationsrate	0.4

Tabelle 3: Parameter für unterschiedliche Metaheuristiken am Beispiel von nrp-e1

empirischem Weg festgelegt, sodass der Algorithmus minimale bis zu maximale Werten beider Zielfunktionen durchläuft. Die Kreuzungsrate bestimmt die Frequenz des Kreuzungsoperators. Es ist nützlich, gleich am Anfang der Optimierung eine vielversprechende Region zu finden. Eine kleine Kreuzungsfrequenz senkt die Wahrscheinlichkeit, schnell auf eine solche Region zu stoßen. Der Mutationsoperator wird von der Mutationsrate kontrolliert, wobei als Mutationsoperator FlipOrExchage-Operator benutzt wurde. Eine hohe Mutationsrate erzeugt eine hohe Diversität der Population. Dadurch kann aber eine Instabilität in der Population ausgelöst werden, sodass sich der Algorithmus wie eine Random Search verhält. Andererseits erweist es sich oft als schwierig, für GA ein globales Optimum mit einer zu kleinen Mutationsrate zu finden, da es ständig zu einem hohen Grad der Intensivierung in dem aktuellen Suchbereich kommt und der Algorithmus nicht die ganze POLM berechnet, wie es z. B. in der Abbildung 28 bei GA zu sehen ist. Nichtsdestotrotz ist GA sehr robust in Bezug auf die Parameterwahl, sodass sich kleine Änderungen der Parameter kaum auf die Qualität der Lösungen auswirken.

Ameisenkolonie: Die zweite Zeile beschreibt die Parameterwahl bei dem Ameisenkolonie-Algorithmus. Sie enthält fünf Parameter: Anzahl der Ameisen 500, α 10, β 10, ρ 0.1 sowie Q 10500. Die Anzahl der Ameisen sollte mindestens auf die Problemgröße gesetzt sein, damit eine positive Rückkoppelung eintreten kann [11]. Im Fall von nrp-e1, wie in der Tabelle 2 zu sehen, sollte die Anzahl der Ameisen mindestens 536 betragen. Im Allgemeinen erwies sich der Ameisenalgorithmus für die Anzahl der Ameisen als nicht sehr empfindlich, solange die Anzahl ausreichend groß war, d. h. über der Problemgröße lag. α und β sind Parameter, welche die Bedeutung der Pheromonpfade sowie der Distanzen zwischen den Knoten darstellen. In dieser Arbeit wurden diese Parameter auf 10 und 10 gesetzt, sodass die Pheromonpfade sowie Distanzen gleich wichtig von Ameisen wahrgenommen werden. ρ ist der Faktor, um welchen die Pfade verdampfen,

und soll aus Erfahrung zwischen 0.1 und 0.01 liegen, weil ansonsten die Qualität der gefundenen Pfade darunter leidet. Q sollte gleich dem optimalen Wert, welcher von dem Problem erwartet wird, gesetzt sein. Im Fall von nrp-e1 liegt der beste gefundene Wert im Bereich von 10500, wie der Tabelle 6 oder Abbildung 33 entnommen werden kann. Q-Parameter, die deutlich kleiner bzw. deutlich größer als der optimale Wert eingesetzt wurden, hatten eine schlechtere Leistung des Algorithmus zur Folge. Der Startpunkt der Ameise wird typischerweise zufällig gewählt, da gezeigt wurde, dass der Startpunkt keinen signifikanten Unterschied ausmacht [11].

Simulated Annealing: Die dritte Zeile enthält die Beschreibung für SA mit vier Parametern: Starttemperatur 1000, Endtemperatur 1, Abkühlungsrate 0.0001 und Mutationsrate 0.5. Die Start- und Endtemperatur entscheiden über die Anzahl der Iterationen. Deswegen hat sich aus Erfahrung der Anfangswert von 1000 Grad adaptiert, sodass es zu einem kontinuierlichen Wechsel von Random Search zu Hill Climbing kommt. Möchte man den Algorithmus länger laufen lassen, so sollte nicht die Starttemperatur erhöht, sondern die Abkühlungsrate verringert werden. In Abhängigkeit von dem Problem liegt die Abkühlungsrate zwischen 0.0001 und 0.00001. Die Mutationsrate hat die gleiche Wirkung wie bei GA. Bei den Metaheuristiken, wo die Anzahl der Iterationen direkt über die Parameter festgelegt werden kann, ist es relativ einfach, die Größenordnung für die Anzahl der Iterationen zu bestimmen. Im Vergleich dazu, steht die Temperatur bei SA in keinem direkten Zusammenhang zu der Anzahl der Iterationen, was oft die Festlegung der Dauer der Ausführung umständlich macht.

Tabu Search: Die letzte Zeile beschreibt die Parameterwahl bei Tabu Search. Sie enthält vier Parameter: Anzahl der Iterationen 1000, Anzahl der Nachbarn 100, die Größe der Tabu-Liste 100 sowie Mutationsrate 0.4. Die Anzahl der Iterationen bei TS hat die gleiche Wirkung wie bei GA und legt fest wie lange der Algorithmus läuft, dabei hat sich der Wert von 1000 durchgesetzt. Die Anzahl der Nachbarn entscheidet wie intensiv der aktuelle Bereich untersucht wird. Aus Erfahrung hat sich der Wert aus dem Bereich zwischen 100 und 200 durchgesetzt. Multipliziert man die beiden Zahlen aus, so wird auf die Größenordnung von 100.000 bis 200.000 Individuen gekommen, die durch TS verarbeitet werden. Das entspricht der Größenordnung des Genetischen Algorithmus. Die Tabu-Liste legt fest, wie viele der letzten Züge bei der nächsten Iteration nicht gewählt werden dürfen, hierbei hat sich der Wert zwischen 100 und 500 als produktiv erwiesen. Bei größerer Tabu-Liste wird der Rechenaufwand merklich erhöht, sodass der Algorithmus länger für jede Iteration benötigt. Die Mutationsrate spielt bei TS die gleiche Rolle wie bei GA.

5 Klassenstruktur in jMetal-Framework

jMetal [30, 36] ist ein objekt-orientiertes Java-basiertes Framework, welches die Entwicklung von Metaheuristiken zum Lösen von Nicht-Pareto- als auch Pareto-Optimierungsproblemen unterstützt. Für diese Arbeit wurde jMetal 5.0 ausgesucht, da das Framework bereits viele Datenstrukturen und Algorithmen in seiner Basisform mitbringt. jMetal liefert eine große Menge an Klassen, die als Grundbausteine bei der Entwicklung weiterer Metaheuristiken genutzt wurden, sodass der Vorteil der Wiederbenutzbarkeit des Codes gewonnen werden konnte. Des Weiteren ist das Framework so konzipiert, dass Probleme, Algorithmen, Operatoren und Lösungen unabhängig voneinander umgesetzt werden, was sich als sehr vorteilhaft erwiesen hatte, weil alle Komponenten nach Bedarf ausgetauscht werden können. Viele Algorithmen verwenden die gleichen Code-Komponenten, wie z. B. genetische Operatoren oder einen Archiv-Operator [40], sodass es dadurch möglich wird, unterschiedliche Metaheuristiken angemessen miteinander zu vergleichen, ohne dass verschiedene Implementierungen von Operatoren einen zu großen Einfluss auf die Leistung der Algorithmen nehmen. Des Weiteren sind in jMetal Mechanismen zur Durchführung von Experimenten integriert. Ein Experiment stellt eine Möglichkeit dar eine Menge von Algorithmen auf unterschiedlichen Problemen laufen zu lassen, sodass Resultate der Ausführung automatisch mittels Metriken ausgewertet werden. Sämtliche Ergebnisse werden anschließend in Form von .csv-Dateien gespeichert. Zur Weiterverwendung der Ergebnisse bietet jMetal die Möglichkeit an, die Ergebnisse zu .tex-Dateien zu konvertieren. Die Boxpots werden automatisch von jMetal generiert. Zur grafischen Darstellung der Resultate dieser Arbeit wurde jMetal um ein weiteres Feature zur Erstellung der Scatterplots erweitert.

Wie in der Abbildung 11 zu sehen ist, bilden vier abstrakte Klassen, Solution, Algorithm, Problem und Operator, die Grundlage für die Entwicklung eigener Komponenten. In der folgenden Auflistung werden diese Strukturen kurz erläutert, bevor zu den konkreten Implementierungen der abstrakten Klassen übergegangen wird:

Problem: Die Problemklasse ist eine Abbildung eines kombinatorischen Problems. Diese Klasse ist in der Lage, Lösungen mittels der Fitnessfunktion zu bewerten.

Solution: Bei der Lösung handelt es sich um eine Datenstruktur, die eine bestimmte an den Algorithmus angepasste Form der Lösung repräsentiert.

Algorithm: Der Algorithmus ist eine Datenstruktur, der alle notwendigen Utilities zur Ausführung des Algorithmus mitbringt, z. B. bietet die Algorithmusklasse eine Möglichkeit zur Erzeugung einer initialen Lösung bzw. Population.

Operator: Die Operatorklasse erlaubt die Umsetzung unterschiedlicher Operatoren, z. B. Mutation, Selektion oder Kreuzung.

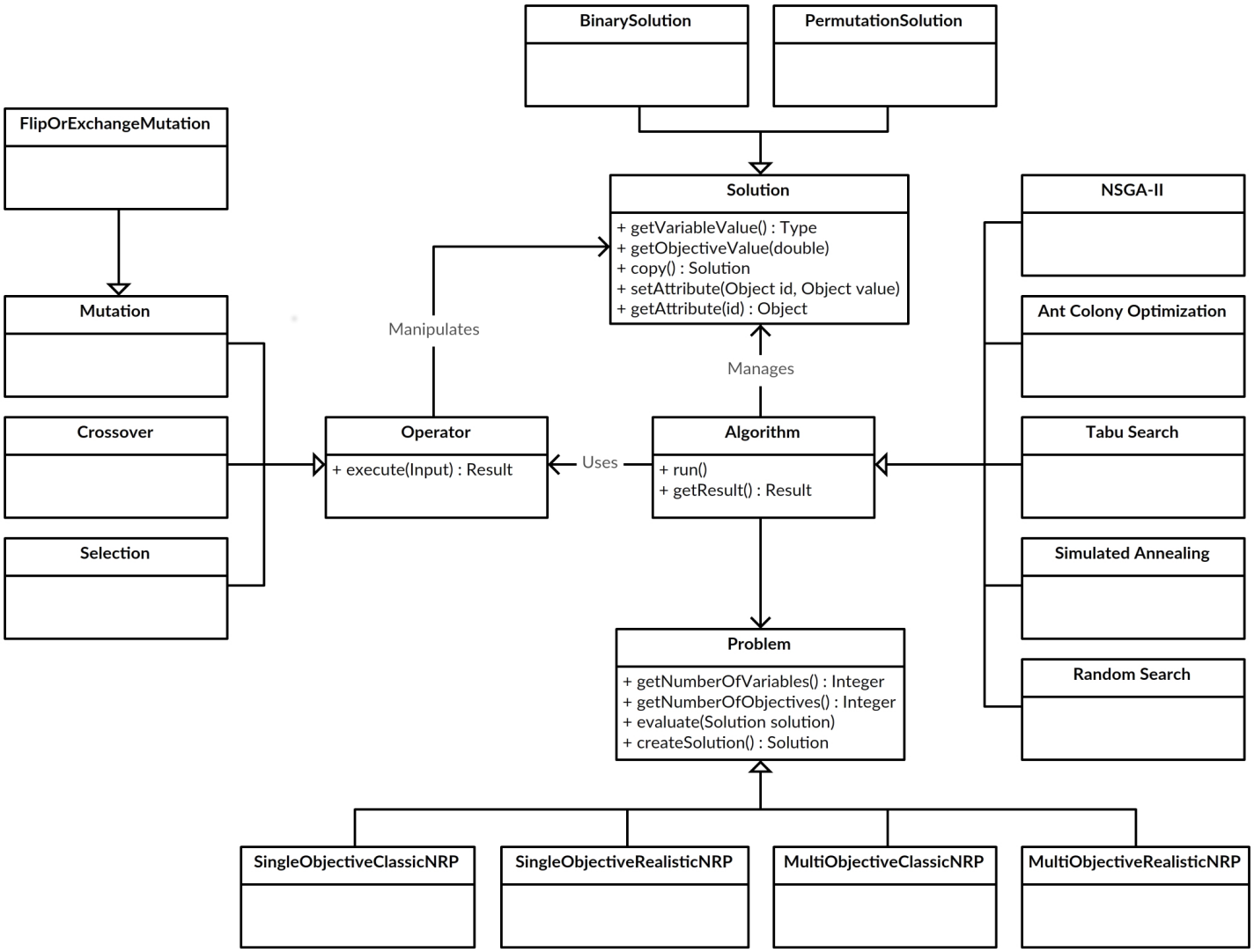


Abbildung 16: Klassendiagramm von grundlegenden Elementen des jMetal-Frameworks

Wie in der Abbildung 16 zu sehen, bildet die Algorithm-Komponente die zentrale Stelle des Programms, indem sie eine Instanz des Problems, der Lösung sowie unterschiedlicher Operatoren beinhaltet. Alle fünf Algorithmen, GA, ACO, TS, SA und RS, wurden aus der abstrakten Klasse Algorithm abgeleitet. Diese Komponente besitzt eine run()-Methode, welche die Ausführung des Algorithmus startet. Am Ende der Ausführung wird im Nicht-Pareto-Fall eine Lösung und im Pareto-Fall eine Liste an Pareto-optimalen Lösungen mittels der Methode getResult() an das Hauptprogramm zurückgegeben. Anschließend erfolgt eine weitere Auswertung mittels Metriken. Der Algorithmus besitzt Lösungsinstanzen, die durch Operatoren verändert und von Problemklassen evaluiert werden. Während der Bearbeitung der Aufgabe wurden mehrere Repräsentationsformen der Lösungen ausprobiert, nämlich PermutationSolution und BinarySolution. Bei PermutationSolution wird die Lösung in Form eines Double-Arrays gespeichert, im BinarySolution Fall wird Java-BitSet genutzt, um die gesetzten Bits der Lösung zu speichern. Die BitSet-Implementierung hat den Vorteil der minimalen Darstellung der gesetzten Bits. Wie groß der Einfluss der Darstellungsform der Lösung auf die Leistung der Algorithmen ist, wurde nicht gemessen, da es nicht im Fokus dieser Arbeit liegt. Die BinarySolution wurde letztendlich zur Darstellungsform der Lösungen in der Implementierung der Metaheuristiken gewählt, weil sie sich unter anderem als besser kompatibel zu anderen Komponenten, wie z. B. der Experiment-Klasse, erwiesen hatte.

Wie bereits in Kapitel 3 über die Datensätze beschrieben, besteht der Hauptunterschied zwischen den klassischen und realistischen Datensätzen darin, dass die ersten Abhängigkeiten zwischen den Anforderungen aufweisen und die zweiten nicht. Aus diesem Grund sind vier Problemklassen entstanden: SingleObjectiveClassicNRP und SingleObjectiveRealisticNRP für klassische und realistische Datensätze mit einer Zielfunktion sowie MultiObjectiveClassicNRP und MultiObjectiveRealisticNRP für klassische und realistische Datensätze mit zwei Zielfunktionen. All diese Klassen wurden aus der abstrakten Problem-Klasse abgeleitet.

Es war während der eigenen Implementierung notwendig, die initiale Lösung der Algorithmen vor der Ausführung zu konfigurieren, da z. B. bei populationsbasierten Verfahren eine Population, welche ausschließlich aus Lösungen mit dem Fitnesswert 0 besteht, eine sehr schlechte Konvergenz gegen ein Optimum zeigte, weil nicht ausreichend unterschiedliche Individuen für die Kreuzung und Selektion in der Population vorhanden waren. Die Konfigurationsmöglichkeit der initialen Lösung hat es ermöglicht, eine initiale Population mit unterschiedlichen Individuen zu erstellen, welche ausreichende Diversität besaßen, aber trotzdem Fitnesswerte nahe bei 0 hatten. Die lokalen Suchalgorithmen wurden immer mit einer initialen Lösung mit dem Fitnesswert 0 gestartet, d. h., alle Bits dieser Lösung waren vor der Ausführung nicht gesetzt.

Das Projekt der Implementierung dieser Arbeit wurde auf GitHub[1] veröffentlicht.

[1] `https://github.com/TarasGit/jMetal`

6 Evaluierung

Dieses Kapitel definiert den Begriff der Leistung und präsentiert Metriken sowie statistische Untersuchungsmethoden, welche bei der Evaluierung der Algorithmen angewandt wurden.

6.1 Leistung der Algorithmen

Da diese Arbeit sowohl Nicht-Pareto- als auch Pareto-Algorithmen behandelt, muss auch zwischen der Leistung der Algorithmen für diese beiden Fälle unterschieden werden. Im Nicht-Pareto-Fall beschriebt die Leistung die Qualität der Lösungen sowie die zur Berechnung benötigte Zeit. In dieser Arbeit wird die Qualität der Nicht-Pareto-Algorithmen wie in der wissenschaftlichen Arbeit von Xuan [2] anhand der Mittelwerte bestimmt, wobei auch maximale Werte zu jedem Algorithmus berechnet werden. Somit definiert diese Arbeit den Begriff der Qualität für Nicht-Pareto-Algorithmen wie folgt:

Definition 10: Qualität für Nicht-Pareto-Algorithmen

Bei Nicht-Pareto-Algorithmen wird die Qualität anhand des Mittelwertes des Gewinns ermittelt.

Im Pareto-Fall ist es schwieriger, über die Leistungsfähigkeit der Algorithmen zu urteilen, da mehrere Zielfunktionen gleichzeitig optimiert werden. Wie bereits im Kapitel 2 definiert, wird die Menge der Lösungen, die von einem bestimmten Algorithmus errechnet wurde, durch Pareto-optimale Lösungsmenge (POLM) dargestellt. Die Pareto-Front gibt die Menge aller Pareto-optimalen Lösungen eines Problems wieder und ist für die vorliegenden NRP-Datensätze unbekannt. Zu ihrer Bestimmung müssen alle Lösungen im Suchraum deterministisch überprüft werden, was aufgrund der NP-Härte des Problems in überschaubarer Zeit unmöglich erscheint. Es wird stattdessen eine Referenz-Front künstlich konstruiert. Sie wird nach der Durchführung der Experimente als Zusammenfassung der POLMs aller Algorithmen zu einem bestimmten Datensatz berechnet. Im Anschluss werden erneut die dominierten Lösungen aussortiert, welche möglicherweise durch die Zusammenfassung entstanden sind.

Ein Algorithmus kann zu der Referenz-Front beitragen oder Lösungen produzieren, die von den Lösungen anderer Algorithmen dominiert werden. Somit existiert ein Bedürfnis, die Leistungsfähigkeit der Algorithmen anhand der berechneten Pareto-optimalen Lösungsmenge zu messen. Metriken werden dazu verwendet, um die Leistungsfähigkeit der Algorithmen in Zahlen auszudrücken. Aber ungeachtet aller Entwicklungen und einer großen Anzahl an Metriken existieren keine offiziellen Standardangaben, welche Metriken benutzt werden sollen. Zur Klärung des Begriffs der Leistungsfähigkeit für Pareto-Algorithmen greift diese Arbeit auf die Definition von Zhang [2] zurück. In seiner Arbeit misst er die Leistung der Metaheuristiken anhand von drei Kriterien: Qualität, Diversität und Zeit. Unter Qualität versteht er die Messung des Beitrags (engl. contribution),

einzigartigen Beitrags (engl. unique contribution), der Konvergenz (engl. convergence) sowie des Hypervolumens (engl. hypervolume). Im weiteren Verlauf dieser Arbeit werden englische sowie deutsche Bezeichnungen dieser Kriterien in Abhängigkeit vom Kontext verwendet. Die oben angeführten Leistungskriterien werden im Folgenden kurz erläutert:

Contribution Der Beitrag misst die Anzahl an Lösungen, welche von einem Algorithmus produziert wurden und auf der Referenz-Front liegen.

Unique Contribution Der einzigartige Beitrag liefert eine Anzahl an einzigartigen Lösungen, die von dem Algorithmus erzeugt wurden und auf der Referenz-Front liegen. Einzigartig bedeutet in diesem Kontext, dass diese Lösungen von keinem anderem Algorithmus gefunden wurden. Dabei wird folgender Umstand in Betracht gezogen, dass ein Algorithmus nur wenige Lösungen liefern kann, welche auf der Referenz-Front liefern, aber dennoch von Bedeutung ist, da diese Lösungen einzigartig sind.

Convergence: Für einen Algorithmus bildet die Konvergenz-Metrik die euklidische Distanz zwischen der Referenz-Front und der Pareto-optimalen Lösungsmenge, die von diesem Algorithmus erzeugt wurde. Zur Messung der Konvergenz wurde in dieser Arbeit auf die Generational Distance Metrik zurückgegriffen.

Hypervolume Das Hypervolumen ist das Volumen, welches von den Lösungen im Suchraum eingenommen wird. Das Hypervolumen bildet die Vereinigung der Volumen aller Lösungen der Pareto-optimalen Lösungsmenge.

Diversity: Die Diversität misst das Ausmaß der Streuung der Pareto-optimalen Lösungsmenge. Zur Messung der Diversität wurde in dieser Arbeit die Spread-Metrik genutzt.

Time: Die Zeit-Metrik misst die Wall-Clock-Time, welche zur Erzeugung der Lösungsmenge notwendig war. Alle Experimente wurden auf Ubuntu 16.04 64-Bit Intel Core i5 4 x 2.9 GHz, 8 GB RAM sowie 4 MB L3-Cache durchgeführt.

Die Qualität der Lösungen, gemessen an den vier oben angeführten Kriterien, hat eine höhere Priorität als die Diversität und Zeit, bis eine ausreichende Qualität erreicht ist. Somit wird der Begriff der Leistung für Pareto-Algorithmen in dieser Arbeit wie folgt definiert:

Definition 11: Leistung für Pareto-Algorithmen

Die Leistung bei Pareto-Algorithmen beinhaltet die Messung der Qualität, Diversität und Zeit [2].

Die Leistungskriterien für Pareto-Algorithmen können nicht adäquat mit nur einer Metrik gemessen werden. Im nächsten Abschnitt werden mehrere Metriken beschrieben, welche zur Messung der oben angeführten Kriterien benutzt werden.

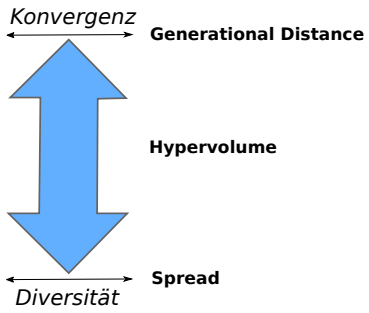


Abbildung 17: Klassifizierung der Metriken [31]

6.2 Metriken

Um die einzelnen Kriterien der Leistung messen zu können, wird sich der Leistungsindikatoren, auch Metriken genannt, bedient. Dabei ist zwischen unären und binären Metriken zu differenzieren. Diese Arbeit nutzt unäre Metriken: Hypervolume sowie Time, um die Leistung ohne Bezug zu einer Referenz-Front zu messen. Als Alternative werden auch binäre Metriken, Generational Distance, Spread, Contribution, Unique Contribution, für den direkten Vergleich zwischen der Pareto-optimalen Lösungsmenge und einer Referenz-Front herangezogen. Eine Kombination aus unären und binären Metriken ermöglicht einen besseren Vergleich der Leistung der Algorithmen. Bei der Implementierung in jMetal wurden alle sechs Leistungskriterien umgesetzt. Die Messung des Beitrags und des einzigartigen Beitrags wird wie folgt durchgeführt. Der Beitrag des Algorithmus wird als Verhältnis der gefundenen Lösungen, welche auf der Referenz-Front liegen zu der gesamten Anzahl der Lösungen der Referenz-Front gemessen. Der einzigartige Beitrag wird durch das Verhältnis der gefundenen Lösungen, die auf der Referenz-Front liegen und von keinem anderen Algorithmus gefunden wurden, zu der Gesamtzahl der Lösungen in der Referenz-Front gebildet. Alle berechneten Fitnesswerte der Lösungen werden in jMetal vor der Anwendung der Metriken normalisiert, sodass sie für beide Zielfunktionen nach der Normalisierung im Bereich zwischen 0 und 1 liegen. Ein Fitnesswert x, von der Menge der berechneten Lösungen, wird im Bereich von x_{min} bis x_{max} normalisiert, indem folgende Normalisierungsformel benutzt wird: $\frac{x-x_{min}}{x_{max}-x_{min}}$. Die Zeit wurde als Wall-Clock-Time in Sekunden gemessen und wurde bei der Auswertung nicht normalisiert.

Die Abbildung 17 beschreibt die Klassifizierung der Metriken in Bezug auf die Diversität sowie Konvergenz [21]. Generational Distance misst die Konvergenz. Spread wird verwendet, um die Diversität zu messen. Das Hypervolumen misst sowohl die Konvergenz als auch Diversität der Daten [31]. Es sollte an dieser Stelle angemerkt werden, dass es auch andere Metriken gibt, um die Konvergenz sowie Diversität zu messen. Zur Messung der Konvergenz stehen u. a. folgende Metriken zur Verfügung: Error Ration, Set Coverage, Generational Distance, Inverted Generational Distance. Zur Messung der Diversität können folgende Metriken benutzt werden: Spacing, Spread [31].

6.2.1 Diversitätsmetrik

Spread [19, 31, 37] wird eingesetzt, um die Diversität der Lösungen zu messen, indem sie die Streuung der Pareto-optimalen Lösungsmenge bestimmt. Spread ist eine Metrik, die die Distanz zwischen den Lösungen der POLM und ihrem Ausmaß misst, um die Leistung des Algorithmus in Bezug auf die Diversität der Lösungen zu bestimmen. Je kleiner der Wert der Spread-Metrik, desto besser ist die Leistung. Folglich erreicht ein Algorithmus die beste Leistung, wenn Spread = 0 ist. Der Wert 0 zeigt, dass eine Pareto-optimale Lösungsmenge erhalten wurde, welche mit der Referenz-Front übereinstimmt. Folgende Formel beschreibt die Berechnung dieser Metrik:

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{|POLM|} |d_i - \bar{d}|}{\sum_{m=1}^{M} d_m^e + |POLM|\bar{d}}$$

Die Variablen in der Formel haben folgende Bedeutung: M stellt die Anzahl der Zielfunktionen dar, |POLM| ist die Anzahl der erhaltenen Lösungen in der Pareto-optimalen Lösungsmenge eines Algorithmus, $\bar{d}_i$ die Distanz zwischen den benachbarten Lösungen, $\bar{d}$ der Mittelwert aller Distanzen und d_m^e ist die Distanz zwischen den Extremwerten der Referenz-Front (RF) und der Pareto-optimalen Lösungsmenge ($POLM$) für die m-te Zielfunktion. In einer idealen POLM beträgt $d_m^e = 0$ und $d_i = \bar{d}$.

Die Abbildung 18 zeigt eine grafische Darstellung der Spread-Metrik im zweidimensionalen Raum. Die theoretische Pareto-Front wird durch die schwarze Linie dargestellt. Die roten Punkte stellen Lösungen aus der Referenz-Front dar. Die gelbe Linie repräsentiert die jeweiligen Distanzen zwischen den Lösungen der Pareto-optimalen Lösungsmenge eines Algorithmus. Ein Nachteil dieser Metrik besteht in der Abhängigkeit von der Referenz-Front.

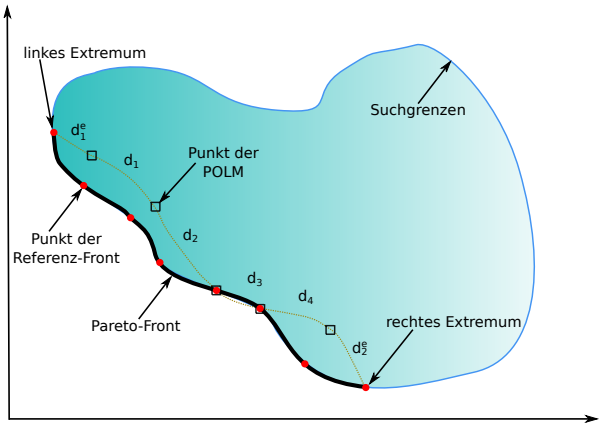


Abbildung 18: Spread-Metrik [19]

6.2.2 Konvergenz-Metrik

Zur Messung der Konvergenz wurde in dieser Arbeit die Generational Distance-Metrik (GD) [31, 37] gewählt. Die Generational Distance wurde von Veldhuizen und Lamont [32] vorgeschlagen, um die euklidische Distanz zwischen der Pareto-optimalen Lösungsmenge und den Lösungen der Referenz-Front zu messen. Folgende Formel präsentiert die Berechnung der GD-Metrik:

$$GD = \sqrt{\frac{\sum_{i=1}^{n} d_i^2}{n}}$$

In der Formel bezeichnet n die Anzahl der Lösungen in POLM, d_i ist die euklidische Distanz zwischen einer Lösung in der POLM und der am nächsten liegenden Lösung der Referenz-Front. GD = 0 bildet den optimalen Wert dieser Metrik. In diesem Fall sind die erhaltene Pareto-optimalen Lösungsmenge und die Referenz-Front identisch und decken sich in allen Punkten. Jeder andere Wert zeigt, dass eine Distanz zwischen der POLM und der Referenz-Front besteht.

Die Abbildung 19 zeigt grafisch die Berechnung der Konvergenz-Metrik. Die Lösungen A bis F stellen die Pareto-optimale Lösungsmenge dar, welche von einem Algorithmus gefunden wurde. Die schwarze Linie bildet die theoretische Pareto-Front. Die roten Punkte bezeichnen die Lösungen der Referenz-Front. Die Linien d_1 bis d_n zeigen Abstände zwischen den sich am nächsten liegenden Punkten.

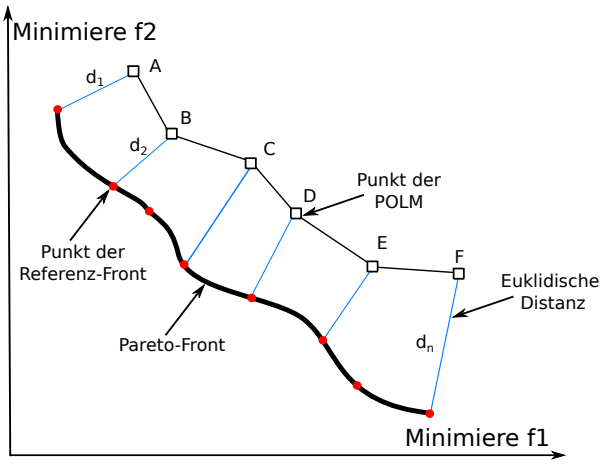


Abbildung 19: Generational Distance-Metrik [19]

6.2.3 Hypervolumen-Metrik

Unter den unären Indikatoren ist der Hypervolumen-Indikator interessant, weil er sowohl die Konvergenz der POLM-Lösungen als auch die Diversität der Lösungen misst [31]. Diese Metrik berechnet das Volumen des Suchraumes, das von der POLM dominiert wird. Die Hypervolumen-Metrik ist streng monoton und der einzige Indikator, der die Pareto-Dominanz in dem Sinne ausdrückt, dass wenn eine Pareto-optimale Lösungsmenge A nur leicht die Pareto-optimale Lösungsmenge B dominiert, diese Dominanz in dem Hypervolumen-Indikator deutlich wird: HVol(A) > HVol(B) [31]. Dabei gilt, dass von zwei Algorithmen derjenige Algorithmus besser ist, welcher einen größeren Wert der Hypervolumen-Metrik aufweist. Je näher die Lösungen der POLM an der Referenz-Front liegen, desto größer ist dabei das Hypervolumen. Zur gleichen Zeit gilt auch, je höher die Streuung der Lösungen, desto höher ist das Hypervolumen. Das Hypervolumen wird mit folgender Formel berechnet:

$$HV = Volume(\cup_{i=1}^{|POLM|} v_i)$$

In der Formel bezeichnet |POLM| die Menge der erhaltenen Pareto-optimalen Lösungen, v_i ist der Hyperwürfel, welcher aus der Lösung $i \in POLM$ und dem Referenz-Punkt ω entsteht.

Ein Vorteil dieser Metrik liegt darin, dass sie unabhängig von der Referenz-Front angewandt und das Hypervolumen direkt aus der POLM berechnet wird. Die Hypervolmen-Metrik hat aber auch einen Nachteil: Die Berechnung dieser Metrik ist NP-hart, sodass die Berechnungszeit exponentiell zu der Anzahl der Zielfunktionen wächst [31]. Die Abbildung 20 beschreibt die Berechnung des Hypervolumen-Indivkators. Die blaue Fläche stellt den Suchraum dar. Die schwarze Linie bildet die Referenz-Front ist aber bei der Berechnung nicht nötig. Das Hypervolumen berechnet die Summe aller Hyperwürfel, welche von den Lösungen A bis E und dem Referenz-Punkt ω gebildet werden.

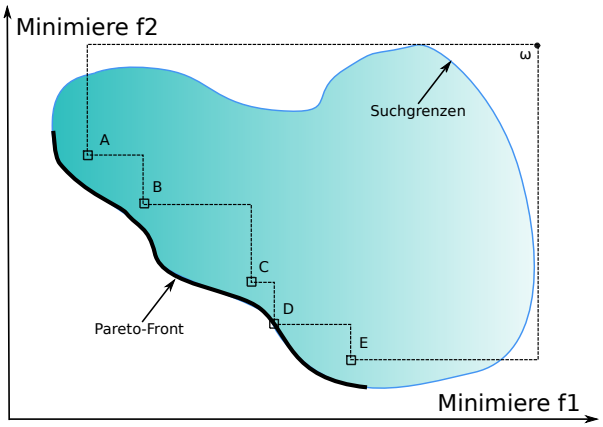


Abbildung 20: Hypervolume-Metrik [31]

6.2.4 Grafische Repräsentation

Viele wissenschaftliche Arbeiten zu Pareto-Algorithmen nutzen oft grafische Plots, um mehrere Referenz-Fronten miteinander zu vergleichen. Die grafische Darstellungsweise hat den Vorteil, dass die Referenz-Front sowie Pareto-optimale Lösungsmengen einzelner Algorithmen grafisch abgebildet werden. Obwohl sich die Interpretation der grafischen Repräsentation der Referenz-Front sowie POLM als eine visuelle Analysemethode als schwierig erweist, wird sie dennoch in vielen wissenschaftlichen Arbeiten, wie z. B. bei Deb [33], angewandt. Auch Zitzler und Thiele [21] nutzen in ihrer Studie die Visualisierung in Form der Scatterplots sowie Boxplots zur Darstellung der Pareto-optimalen Ergebnisse unterschiedlicher Algorithmen. Die visuelle Darstellung wird angewandt, um einen möglichst ausführlichen Vergleich unterschiedlicher Pareto-Algorithmen zu ermöglichen. Sie zeigt unter anderem, ob Metriken adäquat die Spezifika der POLM widerspiegeln. Aus diesem Grund verwendet diese Arbeit grafische Darstellungsformen in Form von Scatterplots in Kombination mit Boxplots, um die Leistung der Algorithmen in unterschiedlichen Experimenten bildhaft und detailliert darzustellen.

6.3 Statistische Untersuchungsmethoden

Bei einem Pareto-Algorithmus entsteht eine Menge an Pareto-optimalen Lösungen als Resultat seiner Ausführung, sodass mehrere Durchläufe notwendig sind, um zuverlässige Ergebnisse zu erhalten. Es ist dabei die stochastische Natur der Algorithmen aufgrund der Randomisierung zu beachten. Die Pseudo-Zufallszahlen, die in Algorithmen benutzt werden, bilden die Ursache für den Nicht-Determinismus. Aus diesem Grund wird in wissenschaftlichen Arbeiten wie Hitchhiker's Guide [34], aber auch bei Zhang [2], empfohlen, statistische Untersuchungsmethoden zu nutzen, um Ergebnisse zu evaluieren.

In den Experimenten wurde die Zahl der Iterationen auf 30 gesetzt, sodass jeder Algorithmus pro Datensatz 30-mal ausgeführt wurde. Zhang lehnt in seiner Arbeit die Anwendung der Mann-Whitney-U-, Wilcoxon- und auch Kruskal-Wallis-Tests ab, weil diese Tests Annahmen über die Verteilung der Testdaten treffen. Er schreibt in seinem wissenschaftlichen Artikel, dass diese nichtparametrisierten Tests von einer konsistenten Varianz in allen Testdaten ausgehen. Des Weiteren legt er dar, dass über die verwendeten Datensätze der NRP-Instanzen keine solche Annahme getroffen werden kann. Deswegen nutzt er in seiner empirischen Arbeit den Vargha-Delaney-$\hat{A}_{12}$ Test zu einem direkten Vergleich der Leistung zweier Algorithmen. Zhang schreibt, dass der Vargha-Deleney-$\hat{A}_{12}$ Test auch einige Annahmen über die Testdaten trifft, aber er geht nicht von einer konsistenten Varianz aus [2]. Vargha-Delaney-$\hat{A}_{12}$ berechnet die Wahrscheinlichkeit, dass sich Algorithmus A besser im direkten Vergleich als Algorithmus B verhält. Aus diesem Grund verwendet auch diese Arbeit die Vargha-Delaney-$\hat{A}_{12}$ Methode, um Algorithmen miteinander zu vergleichen.

Genauso wie hinsichtlich der Qualität, Diversität und Zeit untersucht diese Arbeit auch die Skalierung der Algorithmen, indem die Korrelation zwischen der Problemgröße und der Leistung der Algorithmen gemessen wird. Zhang verwendet in seiner Arbeit statistische Methoden zur Messung der ordinalen Korrelation der Ergebnisse. Er nutzt

die Kendalls-τ und Spearmanns-ρ Rang-Korrelation. Diese beiden Methoden liefern nichtparametrisierte rangbasierte Bewertungen der Korrelation.

Im Folgenden werden Vargha-Delaney-$\hat{A}_{12}$ sowie Rangkorrelationstechniken nach Kendall sowie Spearman kurz erläutert. Bei der Auswertung der Experimente wurde das statistische Programm R verwendet, das die Auswertung der Daten, wie im Anhang beschrieben, vornahm.

6.3.1 Vargha-Delaney-Test

Vargha-Delaney-$\hat{A}_{12}$ ist ein statistischer nichtparametrisierter Test. Im Kontext dieser Arbeit, wenn Daten einer Metrik zu verschiedenen Algorithmen vorliegen, misst $\hat{A}_{12}$ die Wahrscheinlichkeit, dass ein Algorithmus A höhere Metrik-Werte erzielt als ein weiterer Algorithmus B. Diese Größe lässt sich einfacher interpretieren als bei vielen anderen statistischen Tests. Beträgt $\hat{A}_{12} = 0.7$, so verhält sich der Algorithmus A in 70 % der Fälle besser als Algorithmus B. Wenn beide Algorithmen äquivalent sind, so beträgt $\hat{A}_{12} = 0.5$. Außerdem kann dieser Test sehr einfach berechnet werden. Folgende Formel wird im Hitchhiker's Guide [34] angegeben:

$$\hat{A}_{12} = \left(\frac{R_1}{m} - \frac{(m+1)}{2}\right) \cdot \frac{1}{n} \qquad 0 \leq \hat{A}_{12} \leq 1$$

Abbildung 21: Vargha-Delaney-$\hat{A}_{12}$ Berechnungsformel

In der Gleichung bezeichnet R_1 die Summe der Ränge der ersten Datengruppe unter dem Vergleich, m ist die Anzahl der Beobachtungen der ersten Datengruppe, n die Anzahl der Beobachtungen der zweiten Datengruppe. In dieser Arbeit war die Anzahl der Durchläufe für alle Algorithmen gleich, sodass $m = n$ ist.

6.3.2 Rangkorrelation nach Spearman/Kendall

Die Rangkorrelation ist ein nichtparametrisiertes Maß für Korrelation. Sie misst den Zusammenhang zwischen zwei Datengruppen ohne Annahmen über die Verteilung dieser Datengruppen zu machen.

Bei Experimenten sind Fehler nicht auszuschließen. Die Streuung der Daten kann groß sein, sodass Daten in den Experimenten aufgrund einer beschränkten Anzahl der Beobachtungen nicht normalverteilt vorliegen können. Falls davon auszugehen ist, dass die Daten nicht normal verteilt sind, so kann in diesen Fällen kein Testverfahren wie die Korrelation nach Pearson angewandt werden [35]. Anstatt den Korrelationskoeffizienten nach Pearson zu nutzen, werden bei Spearman sowie Kendall im ersten Schritt die Ränge berechnet, die den Ausgangspunkt zur Ermittlung der Rangkorrelationskoeffizienten bilden. Außerdem ist die Rangkorrelation robust gegenüber Ausreißern, da mit Rängen und nicht mit absoluten Werten gearbeitet wird.

Genauso wie die Arbeit von Zhang [2] behandelt auch diese Arbeit zwei Korrelationskoeffizienten nach Spearman und Kendall. Sie sind zwei Rangkorrelationskoeffizienten,

die den Zusammenhang zweier ordinalskalierter Datensätze beschreiben. Beide Verfahren zur Berechnung der Korrelation haben folgende Formeln:

$$\rho_{Sp} = 1 - \frac{6 \cdot \sum_{i=1}^{n} d_i^2}{n \cdot (n^2 - 1)} \qquad -1 \leq \rho_{Sp} \leq +1$$

Abbildung 22: Spearmans-ρ Berechnungsformel [41]

In der Formel bezeichnet d die Rangdifferenzen zwischen zwei Datengruppen, n ist dabei die Anzahl der Beobachtungen, welche bei dem Test verwendet werden.

$$\tau_K = 1 - \frac{4 \sum_{i=1}^{n} q_i}{n \cdot (n-1)} \qquad -1 \leq \tau_K \leq +1$$

Abbildung 23: Kendalls-τ Berechnungsformel [35]

Der Kendall'sche Korreleationskoeffizient nutzt dieselben Rangzahlen wie der Spearman'sche Korrelationskoeffizient. Die Beobachtungen werden nach Rängen geordnet. Die Variable q_i bestimmt für jede Beobachtung i, wie viele Rangzahlen kleiner oder gleich dem aktuellen Rang sind und in der Reihenfolge weiter unten stehen.

Beide Korrelationskoeffizienten können unterschiedliche Werte im Bereich von -1 bis +1 annehmen. -1 bedeutet eine perfekte negative Korrelation. +1 stellt eine perfekte positive Korrelation dar. Sind die Werte nahe bei 0, so liegt gar keine Korrelation vor. Dabei muss in Erinnerung bleiben, dass die Korrelation nicht zwischen den Werten selbst, sondern zwischen ihren Rängen berechnet wird.

7 Experimente

Dieses Kapitel gibt die Resultate der Experimente wieder und ist in zwei Teile gegliedert. Die Experimente sind so ausgelegt, dass im ersten Teil die Nicht-Pareto-Algorithmen auf 17 Datensätzen untersucht werden. Die zwölf klassischen Datensätzen werden mit den Kostenfaktoren 0.3, 0.5 und 0.7 und zwölf realistischen Datensätzen mit den Kostenfaktoren 0.3 und 0.5 ausgeführt [1]. Im zweiten Teil werden die Pareto-Algorithmen auf den gleichen 17 Datensätzen untersucht. Bei Pareto-Problemen wird immer von dem Kostenfaktor 0.5 ausgegangen. Die Experimente für Nicht-Pareto- als auch Pareto-Algorithmen wurden, wie auch in der Arbeit von Zhang [2], 30-mal pro Datensatz ausgeführt.

Der erste Teil beinhaltet die Tabellen 4 bis 7. Die Tabellen 4 und 6 beschreiben die absolute Leistung der Nicht-Pareto-Algorithmen. Dort werden die gemessenen maximalen Werte, Mittelwerte sowie der Mittelwert der zur Berechnung benötigten Zeit zu jedem Datensatz pro Algorithmus angegeben. Die Tabellen 5 und 7 sind aus den Tabellen 4 und 6 abgeleitet. Der durchschnittliche Gewinn wurde benutzt, um die Qualität zu messen, wohingegen maximaler Wert sowie Zeit nur zum Vergleich dienen. Werden die zwei Algorithmen GA und ACO in der Tabelle 5 betrachtet, so wurde folgende Formel wie in der Arbeit von Xuan [1] für die Berechnung der Spalte mit der Bezeichnung „GA“ und „ACO %“ verwendet: $\frac{\omega_{GA}-\omega_{ACO}}{\omega_{ACO}} \cdot 100$. ω_{GA} und ω_{ACO} bezeichnen dabei den durchschnittlichen Gewinn der GA und ACO. Das Resultat der Berechnung stellt den prozentualen Faktor dar, um welchen GA bessere Resultate als ACO liefert. Die Berechnungen für andere Algorithmen erfolgten analog.

Der zweite Teil der Pareto-Algorithmen beinhaltet die Tabellen 8 bis 21. NSGA-II wird in Pareto-Tabellen mit GA abgekürzt. Die Tabellen 8 bis 11 beschreiben den Mittelwert sowie Median zu allen berechneten Metriken. Die dunkelgrau hinterlegten Werte stellen dabei den besten Wert in der gegebenen Zeile dar. Die hellgrau hinterlegten Werte bezeichnen den zweitbesten Wert in der jeweiligen Zeile. Jeder Tabelle 8 bis 11 mit absoluten Daten folgt ein Scatterplot, der die Pareto-optimale Lösungsmenge zu jedem Algorithmus abbildet. Die x-Achse des Scatterplots beschreibt den Gewinn des Kunden und die y-Achse die entstehenden Kosten. Des Weiteren werden sechs Boxplots zu den jeweiligen Metriken nach jedem Scatterplot angegeben, um die Resultate der Berechnung detaillierter darzustellen. Ein Boxplot beschreibt den Median, das obere und untere Quartil sowie Ausreißer bei der Berechnung. Am Boxplot lässt sich auch die Streuung der Daten ablesen, indem die Spannweite zwischen den äußersten Punkten gemessen wird. Die Boxplots werden in dieser Arbeit in zwei Reihen dargestellt. Die erste Reihe, Time, Conv und Diversity, beschreibt Metriken, bei welchen ein kleinerer Metrik-Wert ein besseres Ergebnis darstellt. Die zweite Reihe wird von den Metriken Contrib, UContrib sowie HVol gebildet, bei denen gilt, dass ein größerer Wert der Metrik ein besseres Ergebnis darstellt. Anschließend folgen die Tabellen 12 – 15, die aus den Tabellen 8 – 11 abgeleitet wurden, indem die Vargha-Delaney-$\hat{A}_{12}$ Methode angewendet wurde. Die Tabellen 16 – 21 beschreiben die Rangkorrelation nach Spearman und Kendall. Die Auswertung der Resultate der Experimente anhand der Forschungsfragen erfolgt im nächsten Kapitel.

7.1 Ergebnisse der Nicht-Pareto Algorithmen

Instananz		GA			ACO			TS			SA			RS		
Name	Kostengrenze	Max	Mittelwert	Zeit	Max	Mittelwert	Zeit	Max	Mittelwert	Zeit	Max	Mittelwert	Zeit	Max	Mittelwert	Zeit
nrp-1.0.3	257	1282	1265	6.73	1217	1209	6.42	1282	1281	8.15	1282	1279	21.84	512	353.9	4.58
nrp-1-0.5	429	1932	1925	11.77	1890	1881	15.30	1932	1930	11.18	1932	1920	30.26	1242	978.5	3.90
nrp-1-0.7	600	2526	2516	17.65	2526	2525	15.80	2526	2525	14.70	2526	2522	36.75	1937	1827	4.79
nrp-2-0.3	1514	4872	4745	28.39	3195	2764	30.57	4934	4855	24.53	4974	4915	99.77	1658	1232	6.55
nrp-2-0.5	2524	7679	7484	59.44	5497	5227	82.07	7793	7656	41.44	7875	7798	170.70	2826	2454	5982
nrp-2-0.7	3534	10840	10670	94.82	9005	8685	190.50	10980	10810	62.79	10950	10860	187.20	5596	4682	6.68
nrp-3-0.3	2661	7344	7242	91.81	4844	4520	29.53	7369	7288	58.28	7411	7366	219.30	3384	2916	10.66
nrp-3-0.5	4435	10940	10850	152.20	8590	8037	95.74	10910	10860	98.02	10940	10880	382.80	7112	6469	10.75
nrp-3-0.7	6209	13910	13880	237.20	13270	12930	193.00	13880	13850	131.70	13850	13780	155.80	8619	8380	12.12
nrp-4-0.3	6648	10950	10810	281.90	6235	5913	92.01	11080	11000	173.20	11100	11010	345.20	4324	3703	30.11
nrp-4-0.5	11081	16280	16170	497.50	11670	11330	319.30	16300	16180	304.40	16190	16110	434.50	9385	8694	30.16
nrp-4-0.7	15513	21410	21360	695.50	20490	20290	1169.00	21370	21310	504.70	21340	21250	430.50	12800	12160	39.14
nrp-5-0.3	1198	17880	17530	99.73	9872	9152	77.79	18140	17910	65.69	18320	18140	108.50	6789	6035	9.19
nrp-5-0.5	1996	25240	25030	149.80	19980	19200	330.70	25230	25090	87.03	25220	25050	125700	15610	15110	9.07
nrp-5-0.7	2794	29290	29290	183.00	29290	29290	420.70	29290	29290	107.80	29290	29290	125.30	16190	15860	10.68

Tabelle 4: Leistung der Metaheuristiken bei Nicht-Pareto-Algorithmen für klassische NRP-Instanzen

Datensatz	GA				ACO			TS		SA
	ACO%	TS%	SA%	RS%	TS%	SA%	RS%	SA%	RS%	RS%
nrp-1.0.3	4.63	-1.25	-1.09	258.35	-5.62	-5.47	241.52	0.16	261.86	261.30
nrp-1.0.5	2.34	-0.26	0.26	96.83	-2.54	-2.03	92.33	0.52	97.34	96.32
nrp-1.0.7	-0.35	-0.35	-0.23	37.71	0	0.12	38.20	0.12	38.20	38.04
nrp-2.0.3	71.67	-2.26	0.61	284.14	-43.07	-43.76	124.35	-1.22	294.07	298.94
nrp-2.0.5	43.18	-2.24	-4.02	204.97	-31.73	-32.97	211.98	-1.82	211.98	217.77
nrp-2.0.7	22.85	-1.29	-1.74	127.89	-19.66	-20.03	85.50	-0.46	130.88	131.95
nrp-3.0.3	60.22	-0.63	-1.68	148.35	-37.98	-38.64	55.00	-1.06	149.93	152.61
nrp-3.0.5	35.00	-0.09	-0.27	67.72	-25.99	-26.13	24.24	-0.18	67.88	68.19
nrp-3.0.7	7.35	0.21	0.72	65.63	-6.64	-6.17	54.29	0.51	65.27	64.44
nrp-4.0.3	82.82	1.72	-1.81	191.92	-46.24	-46.29	59.68	-0.09	197.05	197.33
nrp-4.0.5	42.72	-0.06	0.37	85.99	-29.97	-29.67	30.32	0.43	86.10	85.30
nrp-4.0.7	5.27	0.23	0.51	75.65	-4.78	-4.52	66.86	0.28	75.25	74.75
nrp-5.0.3	91.54	-2.12	-3.36	190.47	-48.90	-49.55	51.65	-1.27	196.77	200.58
nrp-5.0.5	30.36	-0.24	-0.08	65.65	-23.47	-23.35	27.07	0.16	66.05	65.78
nrp-5.0.7	0	0	0	84.67	0	0	84.68	0	84.68	84.68

Tabelle 5: Direkter prozentualer Vergleich des Gewinns zwischen zwei Metaheuristiken auf klassischen Nicht-Pareto-Problemen

Instanz		GA			ACO			TS			SA			RS		
Name	Kostengrenze	Max	Mittelwert	Zeit	Max	Mittelwert	Zeit	Max	Mittelwert	Zeit	Max	Mittelwert	Zeit	Max	Mittelwert	Zeit
nrp-e1.0.3	3945	7797	7743	298.00	5415	5209	85.65	7745	7685	127.80	7780	7728	315.60	4348	4038	29.15
nrp-e1-0.5	6575	10990	10950	518.60	8464	8295	214.30	10930	10860	217.50	10900	10850	384.60	7884	7624	28.98
nrp-e2-0.3	4778	7363	7314	444.80	5114	4872	104.20	7334	7276	212.50	7350	7300	471.10	4041	3596	38.95
nrp-e2-0.5	7964	10340	10310	643.10	8070	7723	244.30	10260	10230	281.50	10280	10220	573.50	7238	6901	39.32
nrp-e3-0.3	3120	6579	6534	183.30	4470	4313	48.80	6559	6504	102.60	6560	6517	231.50	3960	3617	22.14
nrp-e3-0.5	5200	9303	9283	294.10	7292	7164	119.80	9260	9215	144.10	9267	9214	307.40	6771	6626	23.14
nrp-e4-0.3	3510	5733	5704	289.40	4073	3941	50.07	5721	5676	115.10	5729	5696	277.30	3541	3048	25.81
nrp-e4-0.5	5850	8142	8103	437.40	6436	6281	138.80	8105	8057	172.90	8097	8042	358.50	5849	5735	27.62
nrp-m1-0.3	4722	10430	10350	262.00	6849	6695	138.00	10470	10380	166.30	10540	10460	440.30	5773	5295	42.32
nrp-m1-0.5	7870	15260	15200	416.30	11390	11060	371.90	15230	15120	247.30	15230	15170	594.70	10380	9615	41.97
nrp-m2-0.3	5099	8507	8435	302.70	5782	5531	130.10	8527	8451	180.50	8545	8468	502.80	4667	4218	51.21
nrp-m2-0.5	8499	12470	12420	581.00	9292	9064	296.10	12400	12350	289.40	12420	12370	604.60	8272	7631	41.95
nrp-m3-0.3	4140	10000	9898	174.20	6715	6496	110.00	10100	10020	144.90	10170	10060	350.70	5825	5486	38.74
nrp-m3-0.5	6900	14720	14660	284.40	11360	11000	331.00	14840	14710	237.10	14820	14750	480.70	10340	9732	37.61
nrp-m4-0.3	4258	7562	7496	172.40	5094	4945	86.61	7607	7553	155.60	7641	7566	334.80	4322	4008	37.70
nrp-m4-0.5	7097	11090	11040	237.60	8555	8403	214.60	11200	11160	239.50	11240	11150	492.10	7917	7517	36.84
nrp-g1-0.3	3983	5996	5959	78.81	4551	4445	50.51	6003	5972	86.93	6012	5984	202.50	4039	3703	16.34
nrp-g1-0.5	6639	8787	8740	141.80	7274	7088	109-10	8783	8727	140.40	8783	8741	252.40	6819	6680	17.89
nrp-g2-0.3	3787	4533	4485	89.68	3397	3334	31.70	4500	4481	84190	4522	4500	178.60	2928	2643	16.97
nrp-g2-0.5	6313	6524	6495	128.20	5494	5372	76.31	6502	6401	140.80	6505	6471	257.40	4904	4824	20.02
nrp-g3-0.3	3677	5824	5791	74.38	4498	4345	39.79	5824	5791	84.95	5864	5811	203.20	3773	3546	16.72
nrp-g3-0.5	6129	8416	8386	120.00	7054	6822	95.21	8396	8353	118.20	8410	8361	232.90	6605	6428	18.23
nrp-g4-0.3	3210	4169	4140	85.53	3167	3098	21.68	4177	4130	67.77	4174	4139	153.10	2671	2478	14.96
nrp-g4-0.5	5350	6031	6000	128.90	5149	4986	54.48	6003	5978	101.90	6018	5983	197.80	4725	4599	14.69

Tabelle 6: Leistung der Metaheuristiken bei Nicht-Pareto-Algorithmen für realistische NRP-Instanzen

Datensatz	GA				ACO			TS		SA
	ACO%	TS%	SA%	RS%	TS%	SA%	RS%	SA%	RS%	RS%
nrp-e1.0.3	48.65	0.75	0.19	91.75	-32.22	-32.59	28.99	-0.56	90.32	91.38
nrp-e1.0.5	32.01	0.83	0.92	43.62	-23.62	-23.55	8.80	0.09	42.44	42.31
nrp-e2.0.3	11.90	0.52	0.19	103.39	-33.04	-33.26	35.48	-0.33	102.34	103.00
nrp-e2.0.5	33.50	0.78	0.88	49.40	-24.51	-24.43	11.91	0.10	48.24	48.10
nrp-e3.0.3	51.49	0.46	0.26	80.65	-33.69	-33.82	19.24	-0.20	79.82	80.18
nrp-e3.0.5	29.58	0.74	0.75	40.09	-22.26	-22.25	8.12	0.01	39.07	39.06
nrp-e4.0.3	44.73	0.49	0.14	87.14	-30.57	-30.81	29.29	-0.35	86.22	86.88
nrp-e4.0.5	29.00	0.57	0.76	41.29	-22.04	-21.90	9.52	0.78	40.49	40.23
nrp-m1.0.3	54.59	-0.29	-1.05	95.47	-35.50	-35.99	26.44	-0.76	96.03	97.54
nrp-m1.0.5	37.43	0.53	0.20	58.08	-26.85	-27.09	15.03	-0.33	57.25	57.77
nrp-m2.0.3	52.50	-0.19	-0.39	99.98	-34.55	-34.68	31.13	-0.20	100.35	100.76
nrp-m2.0.5	37.02	0.56	0.40	62.76	-26.61	-26.72	18.78	-0.16	61.84	62.09
nrp-m3.0.3	55.37	-1.22	-1.61	80.42	-35.19	-35.43	18.41	-0.40	82.65	83.37
nrp-m3.0.5	33.27	-0.34	-0.61	50.64	-25.22	-25.42	13.03	-0.27	51.15	51.56
nrp-m4.0.3	51.58	-0.75	-0.01	87.02	-34.53	-34.64	23.38	-0.17	88.45	88.77
nrp-m4.0.5	31.38	-1.07	-0.99	46.87	-24.70	-24.64	11.78	0.09	48.46	48.33
nrp-g1.0.3	34.06	-0.22	-0.42	60.92	-25.57	-25.72	20.04	-0.20	61.27	61.60
nrp-g1.0.5	23.31	0.15	-0.01	30.84	-18.78	-18.91	6.11	-0.16	30.64	30.85
nrp-g2.0.3	34.52	0.09	-0.33	69.69	-25.60	-25.91	26.14	-0.42	69.54	70.26
nrp-g2.0.5	20.90	1.47	0.37	34.64	-16.07	-16.98	11.36	-1.08	32.69	34.14
nrp-g3.0.3	33.28	0	-0.34	63.31	-24.97	-25.23	22.53	-0.34	63.31	63.87
nrp-g3.0.5	22.92	0.40	0.30	30.46	-18.33	-18.41	6.13	-0.09	29.95	30.07
nrp-g4.0.3	33.63	0.24	0.02	67.07	-24.99	-25.15	25.02	-0.22	66.66	67.03
nrp-g4.0.5	20.34	0.37	0.28	30.46	-16.59	-16.66	8.41	-0.08	29.98	30.10

Tabelle 7: Direkter prozentualer Vergleich des Gewinns zwischen zwei Metaheuristiken auf realistischen Nicht-Pareto-Problemen

7.2 Ergebnisse der Pareto-Algorithmen

7.2.1 Klassischer Datensatz

Datensatz	Metrik		GA		ACO		TS		SA		RS	
			Mittelwert	Median	Mittelwert	Median	Mittelwert	Median	Mittelwert	Median	Mittelwert	Median
nrp1	Qualität	Contrib	9.92e-01	9.92e-01	1.14e-02	1.14e-02	5.27e-01	5.27e-01	5.30e-02	5.30e-02	7.58e-03	7.58e-03
		UContrib	4.73e-01	4.73e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
		Conv	6.16e-05	5.07e-05	8.38e-03	8.52e-03	3.44e-03	2.99e-03	3.13e-02	3.07e-02	5.98e-02	5.97e-02
		HVol	5.55e-01	5.55e-01	4.42e-01	4.41e-01	4.75e-01	4.98e-01	2.59e-01	2.57e-01	1.78e-01	1.78e-01
	Diversity		8.33e-01	8.30e-01	4.99e-01	4.97e-01	5.97e-01	5.74e-01	7.19e-01	7.26e-01	7.08e-01	7.06e-01
	Time		1.43e+01	1.43e+01	5.91e+00	5.95e+00	1.01e+01	1.11e+01	8.25e+00	8.23e+00	1.98e+00	1.98e+00
nrp2	Qualität	Contrib	4.21e-01	4.21e-01	1.67e-03	1.67e-03	5.82e-01	5.82e-01	0.00e+00	0.00e+00	8.35e-04	8.35e-04
		UContrib	4.18e-01	4.18e-01	0.00e+00	0.00e+00	5.78e-01	5.78e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00
		Conv	9.70e-04	9.73e-04	1.16e-02	1.14e-02	1.31e-03	1.33e-03	3.01e-02	3.01e-02	3.82e-02	3.80e-02
		HVol	5.10e-01	5.11e-01	3.13e-01	3.14e-01	4.85e-01	4.85e-01	1.58e-01	1.60e-01	1.35e-01	1.34e-01
	Diversity		5.50e-01	5.54e-01	6.03e-01	6.02e-01	7.93e-01	7.97e-01	7.73e-01	7.69e-01	7.56e-01	7.50e-01
	Time		3.24e+01	3.05e+01	2.23e+02	2.14e+02	1.37e+02	1.33e+02	1.75e+02	1.74e+02	3.29e+01	3.24e+01
nrp3	Qualität	Contrib	1.37e-01	1.37e-01	2.88e-03	2.88e-03	8.66e-01	8.66e-01	0.00e+00	0.00e+00	9.61e-04	9.61e-04
		UContib	1.34e-01	1.34e-01	0.00e+00	0.00e+00	8.60e-01	8.60e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00
		Conv	7.44e-04	7.44e-04	8.67e-03	8.58e-03	3.93e-04	3.59e-04	1.74e-02	1.72e-02	1.93e-02	1.93e-02
		HVol	5.52e-01	5.52e-01	3.55e-01	3.55e-01	5.38e-01	5.47e-01	2.56e-01	2.56e-01	2.44e-01	2.45e-01
	Diversity		3.84e-01	3.82e-01_	5.21e-01	5.21e-01	7.01e-01	6.91e-01	7.69e-01	7.74e-01	7.58e-01	7.60e-01
	Time		5.68e+01	5.64e+01	1.93e+02	1.93e+02	1.13e+02	1.19e+02	3.62e+01	3.57e+01	1.60e+01	1.60e+01
nrp4	Qualität	Contrib	1.40e-01	1.40e-01	1.57e-03	1.57e-03	8.61e-01	8.61e-01	0.00e+00	0.00e+00	3.93e-04	3.93e-04
		UContrib	1.39e-01	1.39e-01	0.00e+00	0.00e+00	8.59e-01	8.59e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00
		Conv	1.62e-03	1.59e-03	9.71e-03	9.76e-03	5.04e-04	4.77e-04	1.86e-02	1.86e-02	2.00e-02	2.01e-02
		HVol	5.43e-01	5.43e-01	3.17e-01	3.16e-01	5.50e-01	5.51e-01	2.19e-01	2.18e-01	2.11e-01	2.11e-01
	Diversity		4.47e-01	4.46e-01	5.46e-01	5.46e-01	8.17e-01	8.13e-01	8.35e-01	8.28e-01	8.03e-01	7.95e-01
	Time		1.31e+02	1.30e+02	8.18e+02	8.15e+02	3.49e+02	3.61e+02	1.17e+02	1.14e+02	4.77e+01	4.62e+01
nrp5	Qualität	Contrib	1.47e-01	1.47e-01	0.00e+00	0.00e+00	8.53e-01	8.53e-01	0.00e+00	0.00e+00	7.43e-04	7.43e-04
		UContrib	1.47e-01	1.47e-01	0.00e+00	0.00e+00	8.52e-01	8.52e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00
		Conv	2.39e-03	2.33e-03	9.47e-03	9.45e-03	9.63e-04	9.63e-04	1.81e-02	1.81e-02	1.87e-02	1.89e-02
		Hvol	5.25e-01	5.25e-01	3.05e-01	3.05e-01	5.36e-01	5.45e-01	2.04e-01	2.04e-01	2.04e-01	2.04e-01
	Diversity		4.73e-01	4.73e-01	4.93e-01	4.89e-01	7.52e-01	7.40e-01	8.29e-01	8.30e-01	7.83e-01	7.82e-01
	Time		5.56e+01	5.54e+01	9.63e+02	9.58e+02	1.11e+02	1.12e+02	3.44e+01	3.36e+01	1.30e+01	1.27e+01

Tabelle 8: Leistung der Algorithmen (Mittelwert/Median) für klassischen Datensatz

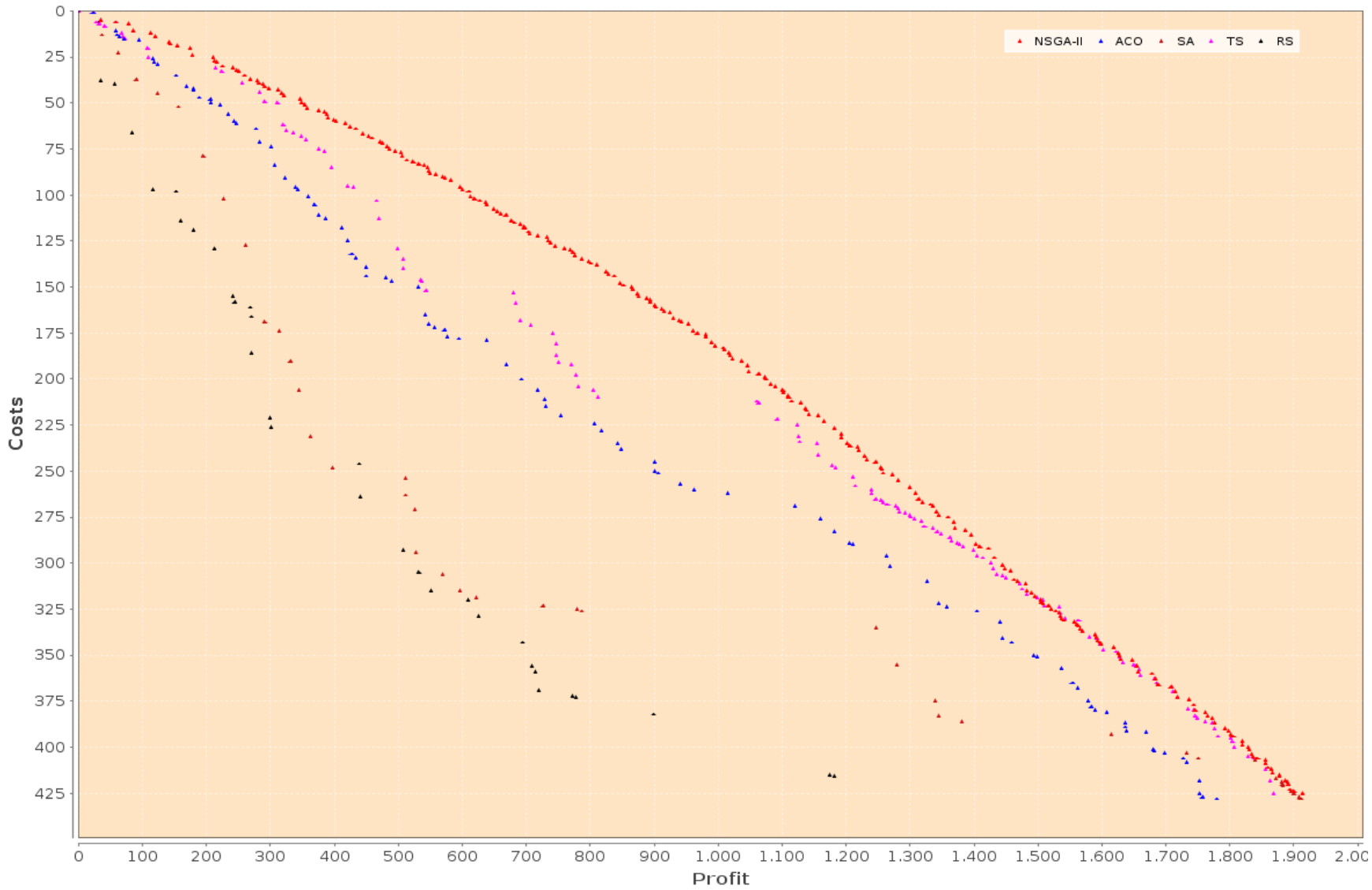
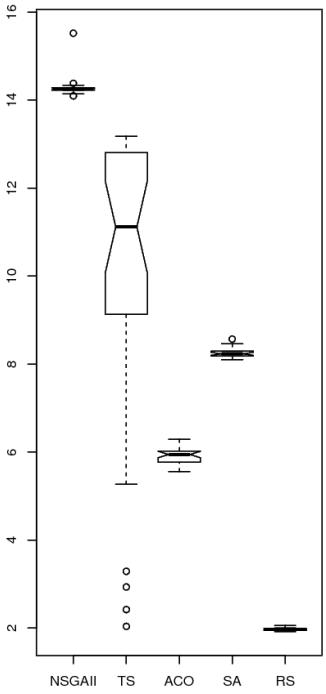
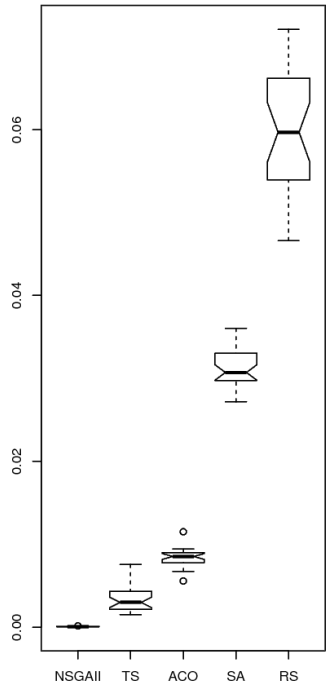


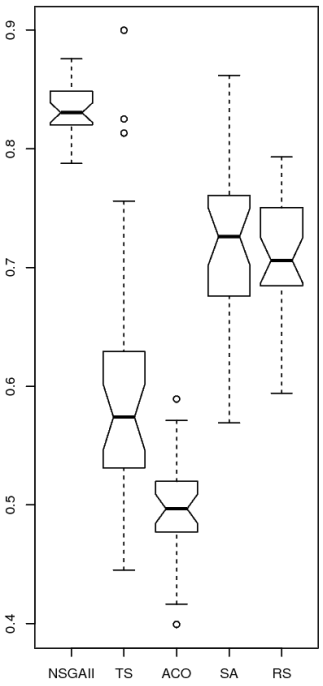
Abbildung 24: Scatterplot für nrp1



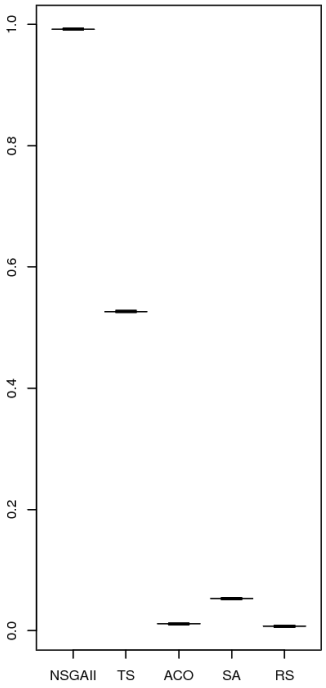
nrp1: Time(s)



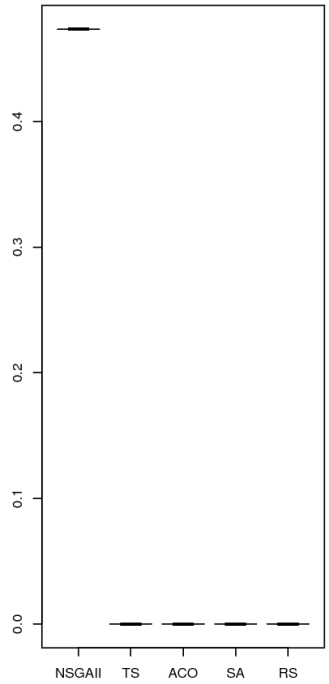
nrp1: Conv



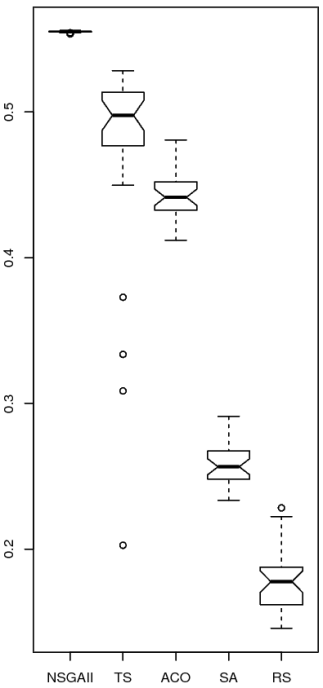
nrp1: Diversity



nrp1: Contrib



nrp1: UContrib



nrp1: HVol

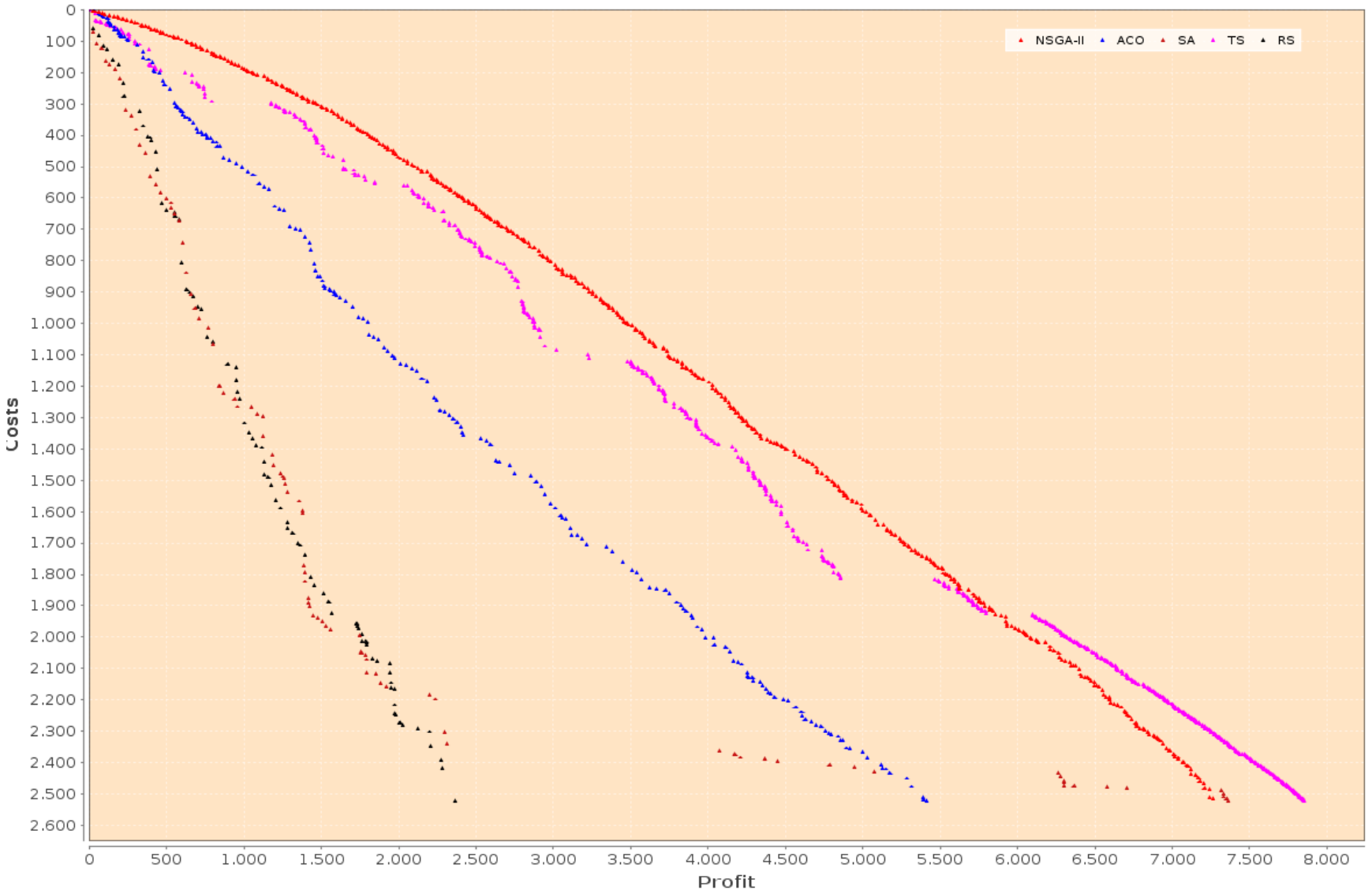
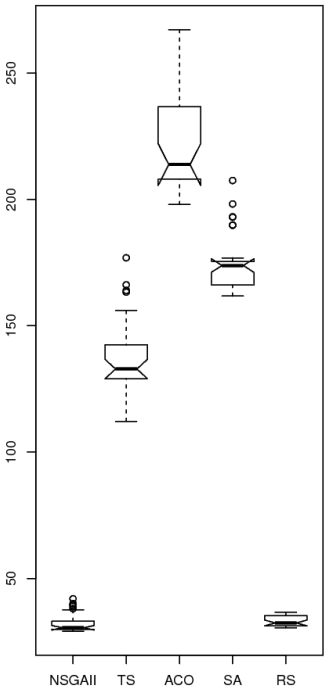
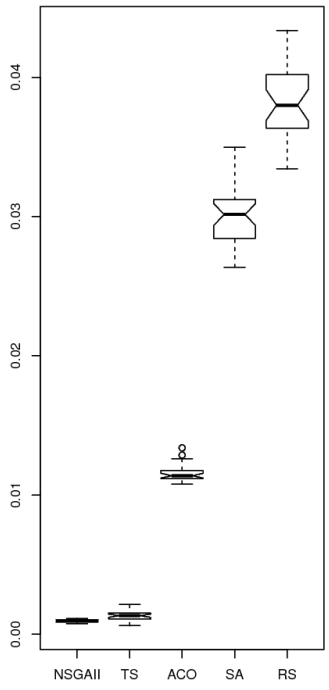


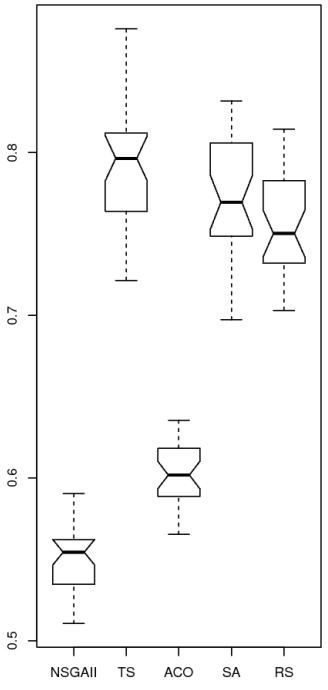
Abbildung 25: Scatterplot für nrp2



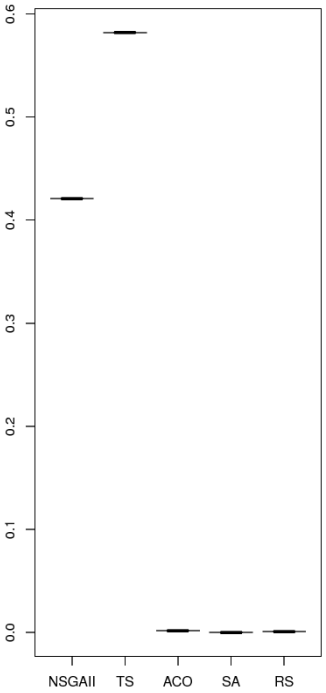
nrp2: Time(s)



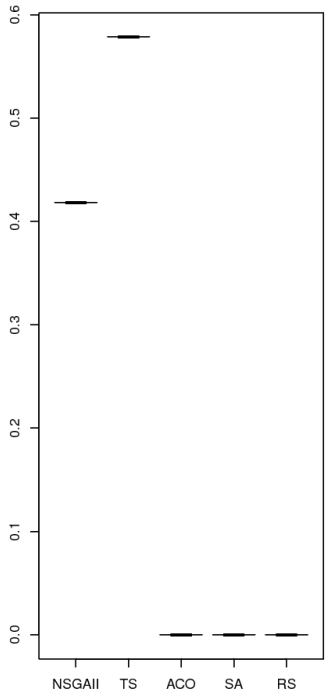
nrp2: Conv



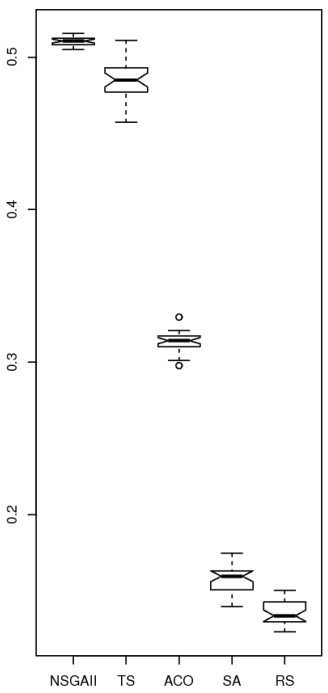
nrp2: Diversity



nrp2: Contrib



nrp2: UContrib



nrp2: HVol

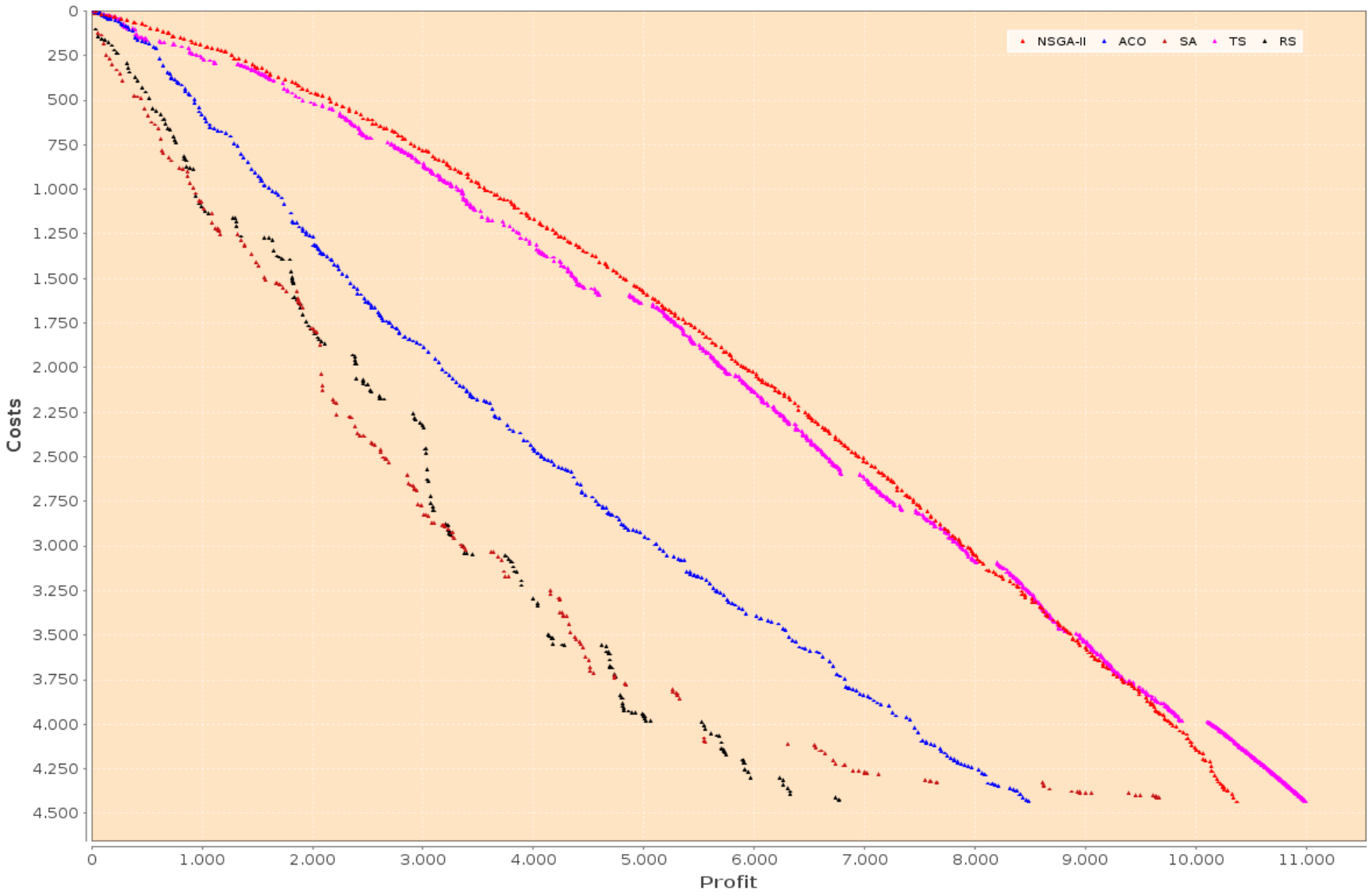
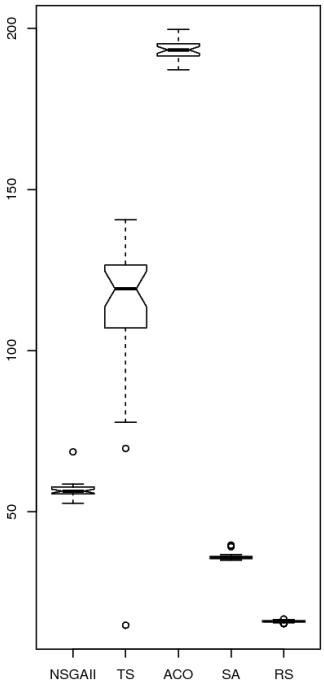
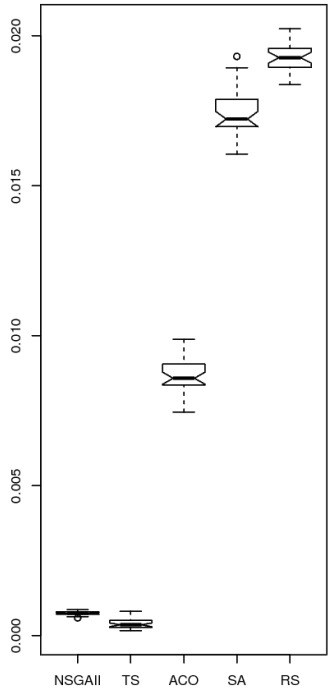


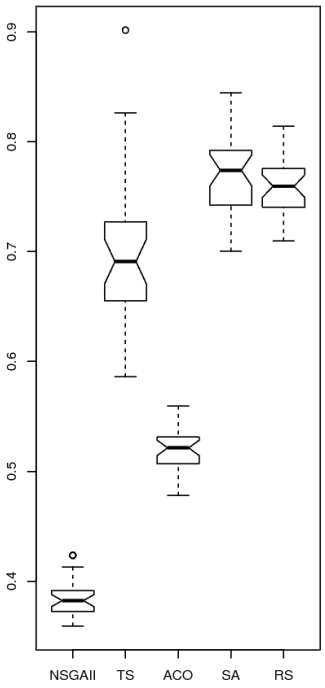
Abbildung 26: Scatterplot für nrp3



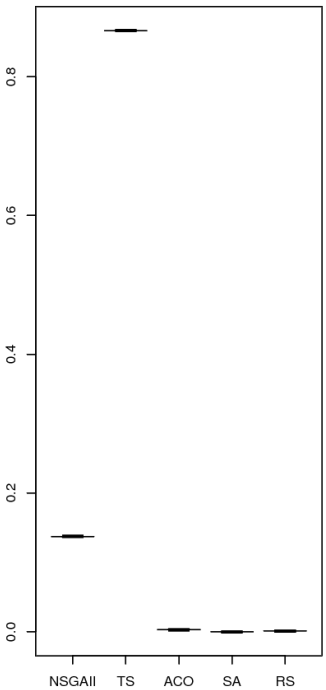
nrp3: Time(s)



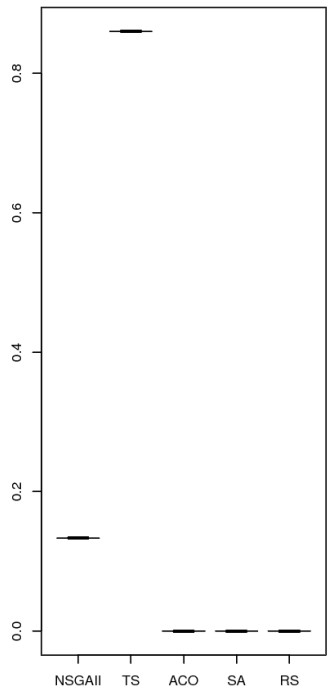
nrp3: Conv



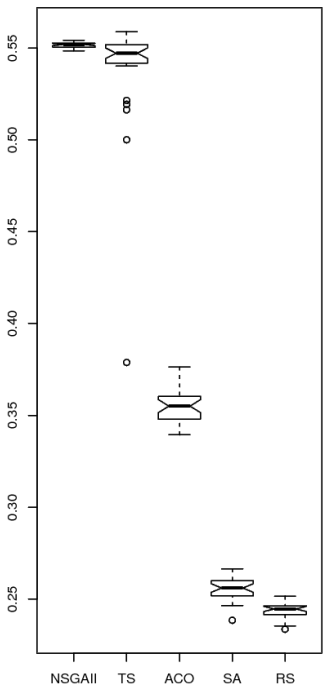
nrp3: Diversity



nrp3: Contrib



nrp3: UContrib



nrp3: HVol

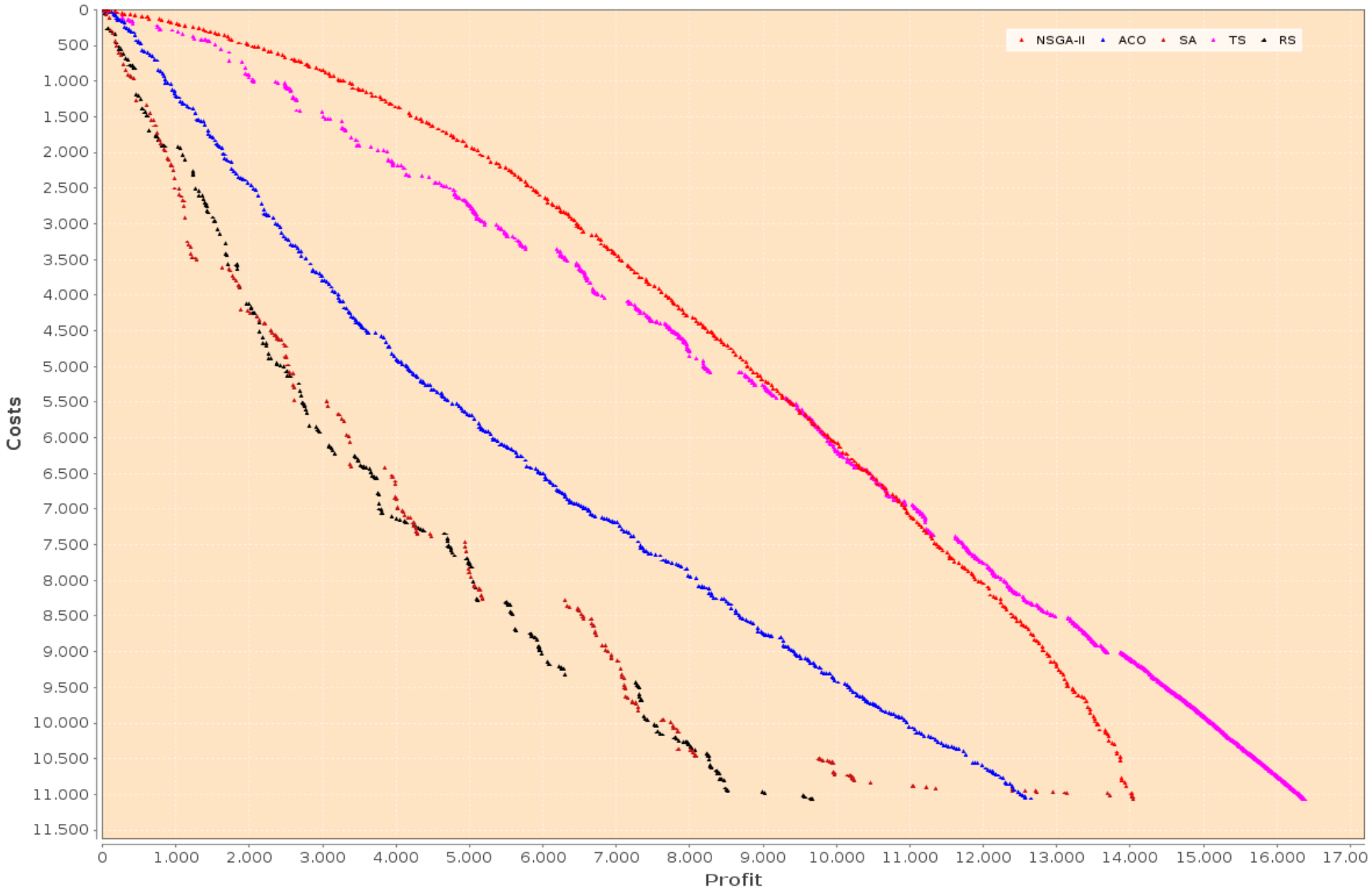
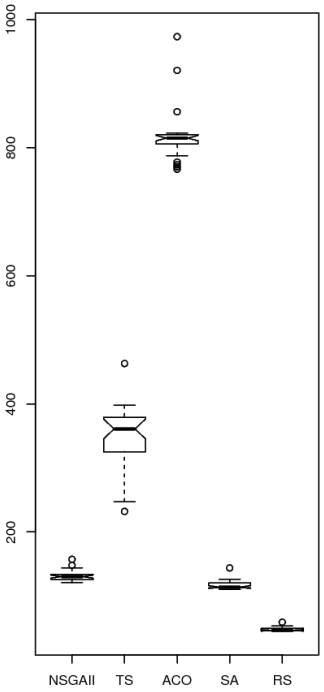
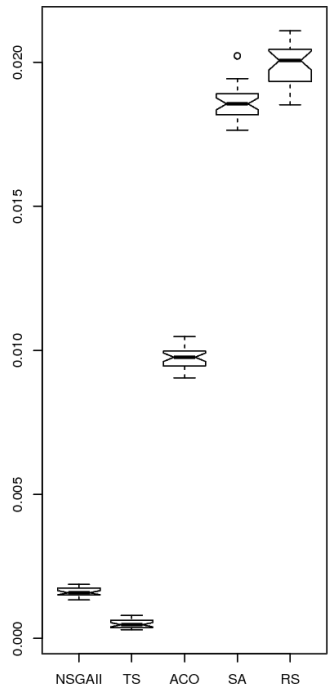


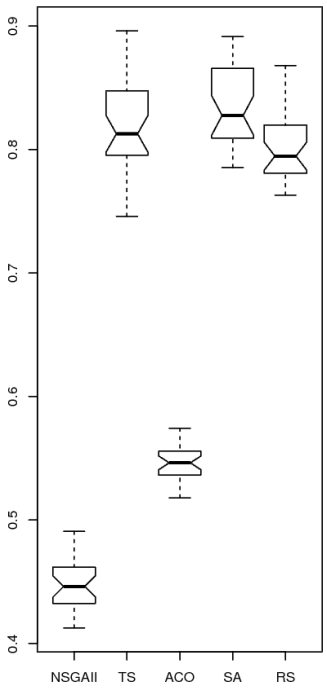
Abbildung 27: Scatterplot für nrp4



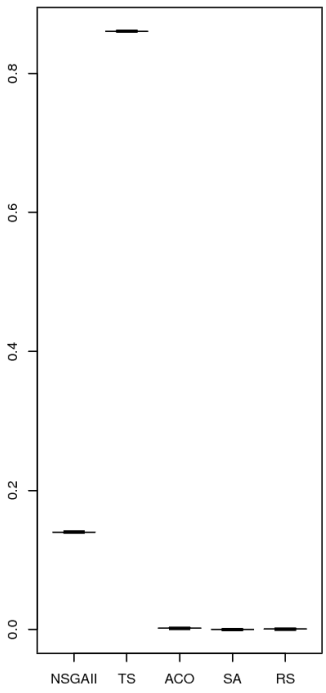
nrp4: Time(s)



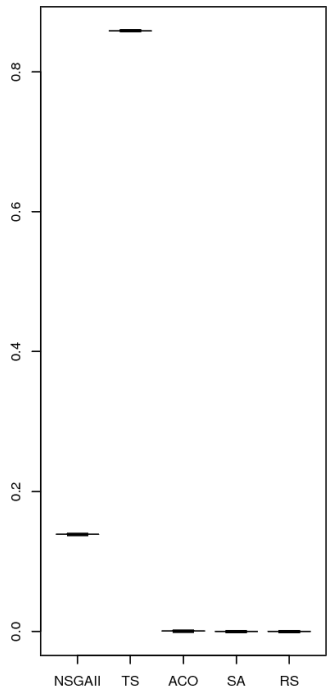
nrp4: Conv



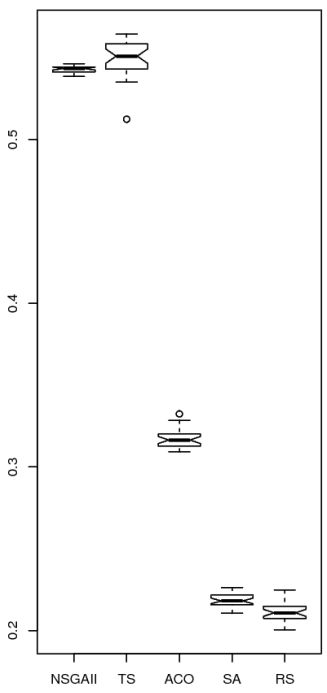
nrp4: Diversity



nrp4: Contrib



nrp4: UContrib



nrp4: HVol

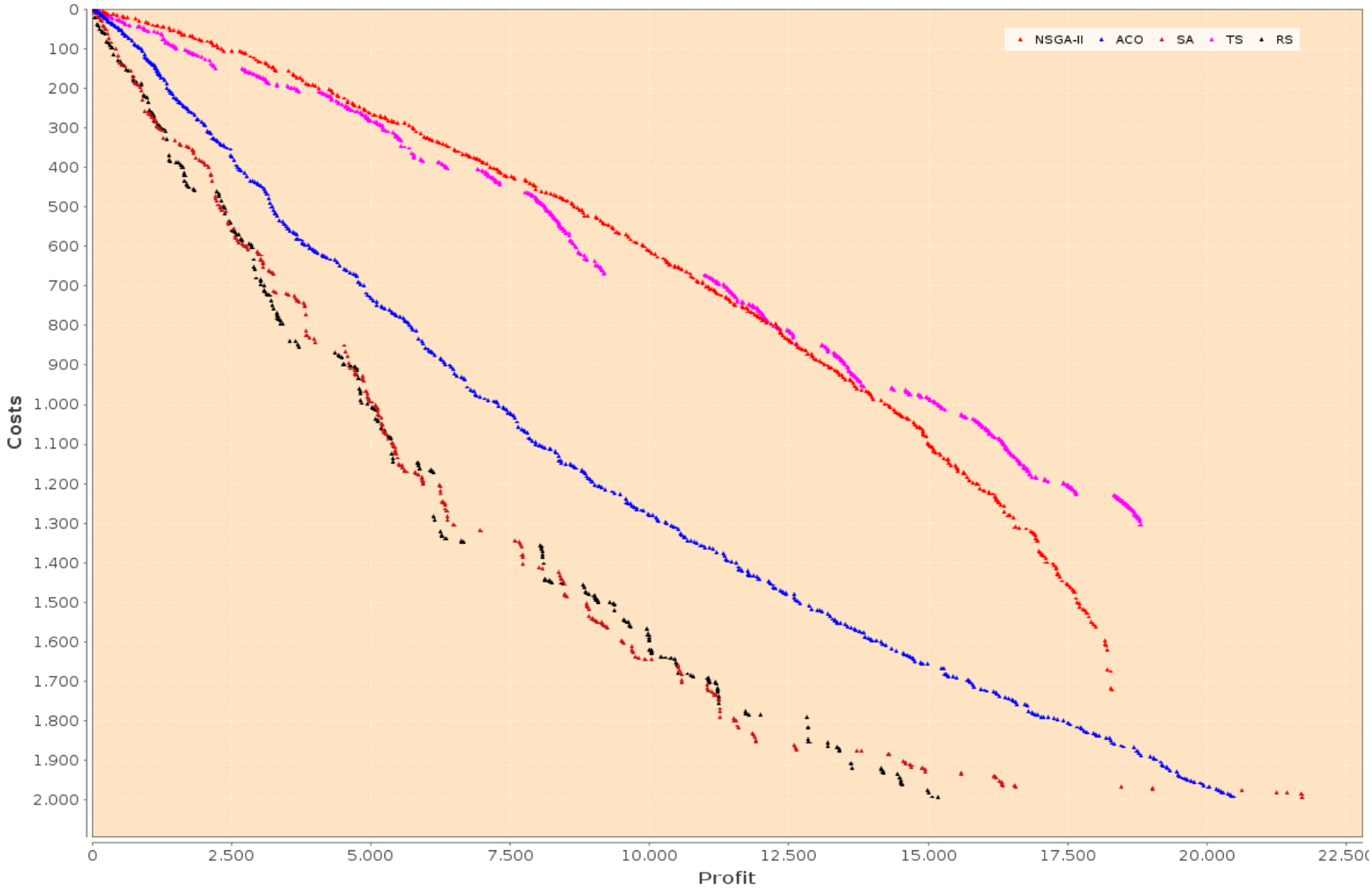
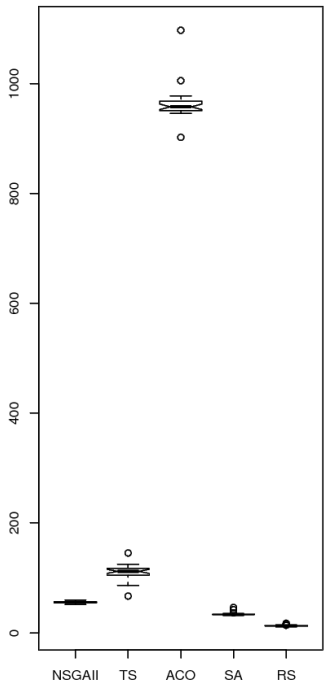
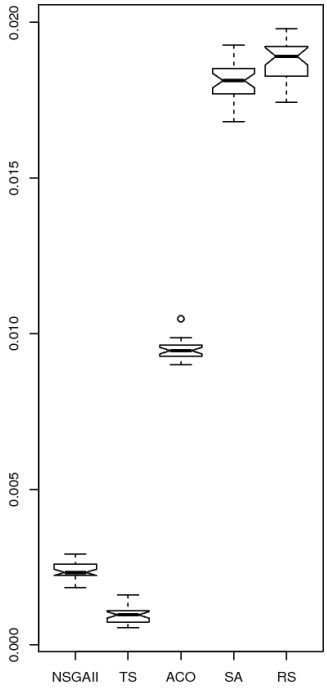


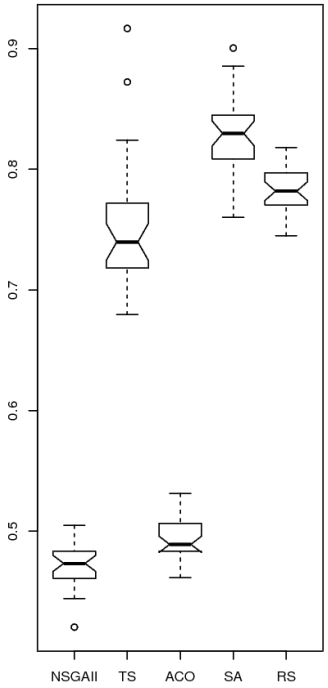
Abbildung 28: Scatterplot für nrp5



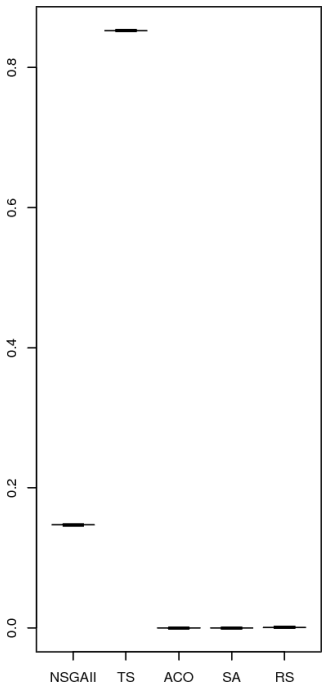
nrp5: Time(s)



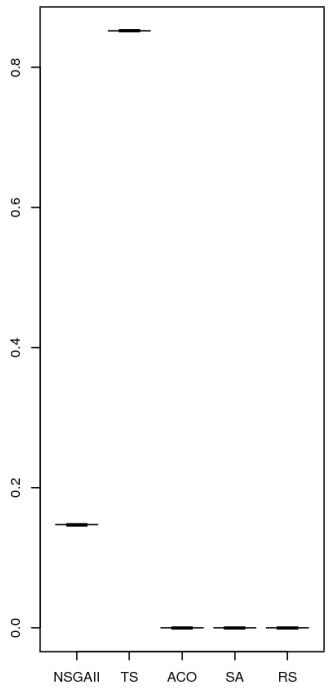
nrp5: Conv



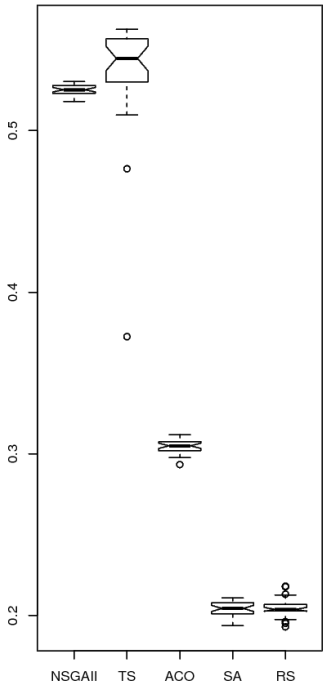
nrp5: Diversity



nrp5: Contrib



nrp5: UContrib



nrp5: HVol

7.2.2 Mozilla-Datensatz

Datensatz	Metrik		GA		ACO		TS		SA		RS	
			Mittelwert	Median	Mittelwert	Median	Mittelwert	Median	Mittelwert	Median	Mittelwert	Median
nrp-m1	Qualität	Contrib	2.08e-01	2.08e-01	3.40e-03	3.40e-03	7.94e-01	7.94e-01	0.00e+00	0.00e+00	4.86e-04	4.86e-04
		UContrib	2.05e-01	2.05e-01	9.73e-04	9.73e-04	7.91e-01	7.91e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00
		Conv	8.45e-04	8.45e-04	5.62e-03	5.69e-03	4.16e-04	3.96e-04	1.10e-02	1.10e-02	1.18e-02	1.19e-02
		HVol	5.49e-01	5.49e-01	3.91e-01	3.92e-01	5.39e-01	5.43e-01	3.01e-01	3.01e-01	2.93e-01	2.94e-01
	Diversity		3.91e-01	3.92e-01	4.94e-01	4.99e-01	6.85e-01	6.83e-01	7.17e-01	7.14e-01	6.67e-01	6.67e-01
	Time		1.92e+02	1.91e+02	6.87e+02	6.86e+02	1.69e+02	1.70e+02	1.69e+02	1.69e+02	5.99e+01	5.95e+01
nrp-m2	Qualität	Contrib	2.58e-01	2.58e-01	2.39e-03	2.39e-03	7.42e-01	7.42e-01	4.78e-04	4.78e-04	4.78e-04	4.78e-04
		UContrib	2.57e-01	2.57e-01	9.57e-04	9.57e-04	7.40e-01	7.40e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00
		Conv	4.80e-04	4.93e-04	6.31e-03	6.34e-03	3.42e-04	3.42e-04	1.27e-02	1.26e-02	1.42e-02	1.42e-02
		HVol	5.59e-01	5.59e-01	3.92e-01	3.92e-01	5.45e-01	5.51e-01	2.97e-01	2.97e-01	2.84e-01	2.84e-01
	Diversity		3.56e-01	3.56e-01	5.07e-01	5.07e-01	6.82e-01	6.74e-01	7.30e-01	7.33e-01	6.84e-01	6.85e-01
	Time		2.35e+02	2.34e+02	5.58e+02	5.51e+02	1.97e+02	2.01e+02	1.88e+02	1.80e+02	6.85e+01	6.58e+01
nrp-m3	Qualität	Contrib	2.22e-01	2.22e-01	4.91e-04	4.91e-04	7.78e-01	7.78e-01	0.00e+00	0.00e+00	4.91e-04	4.91e-04
		UContib	2.22e-01	2.22e-01	0.00e+00	0.00e+00	7.78e-01	7.78e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00
		Conv	8.82e-04	8.75e-04	5.61e-03	5.60e-03	3.85e-04	3.77e-04	1.02e-02	1.03e-02	1.08e-02	1.09e-02
		HVol	5.49e-01	5.49e-01	3.91e-01	3.90e-01	5.32e-01	5.39e-01	3.13e-01	3.14e-01	3.08e-01	3.08e-01
	Diversity		3.84e-01	3.84e-01	4.97e-01	4.99e-01	6.92e-01	6.85e-01	6.76e-01	6.77e-01	6.38e-01	6.40e-01
	Time		1.60e+02	1.60e+02	5.90e+02	5.83e+02	1.31e+02	1.31e+02	1.34e+02	1.30e+02	4.69e+01	4.57e+01
nrp-m4	Qualität	Contrib	2.76e-01	2.76e-01	2.48e-03	2.48e-03	7.25e-01	7.25e-01	0.00e+00	0.00e+00	4.95e-04	4.95e-04
		UContrib	2.74e-01	2.74e-01	0.00e+00	0.00e+00	7.23e-01	7.23e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00
		Conv	3.63e-04	3.67e-04	5.60e-03	5.63e-03	3.18e-04	3.13e-04	1.15e-02	1.14e-02	1.27e-02	1.26e-02
		HVol	5.55e-01	5.55e-01	4.05e-01	4.05e-01	5.33e-01	5.45e-01	3.12e-01	3.11e-01	3.06e-01	3.07e-01
	Diversity		3.52e-01	3.52e-01	5.04e-01	4.99e-01	6.48e-01	6.31e-01	6.83e-01	6.86e-01	6.48e-01	6.51e-01
	Time		2.04e+02	1.98e+02	4.20e+02	4.14e+02	1.59e+02	1.60e+02	1.53e+02	1.47e+02	5.35e+01	5.18e+01

Tabelle 9: Leistung der Algorithmen (Mittelwert/Median) für Mozilla-Datensatz

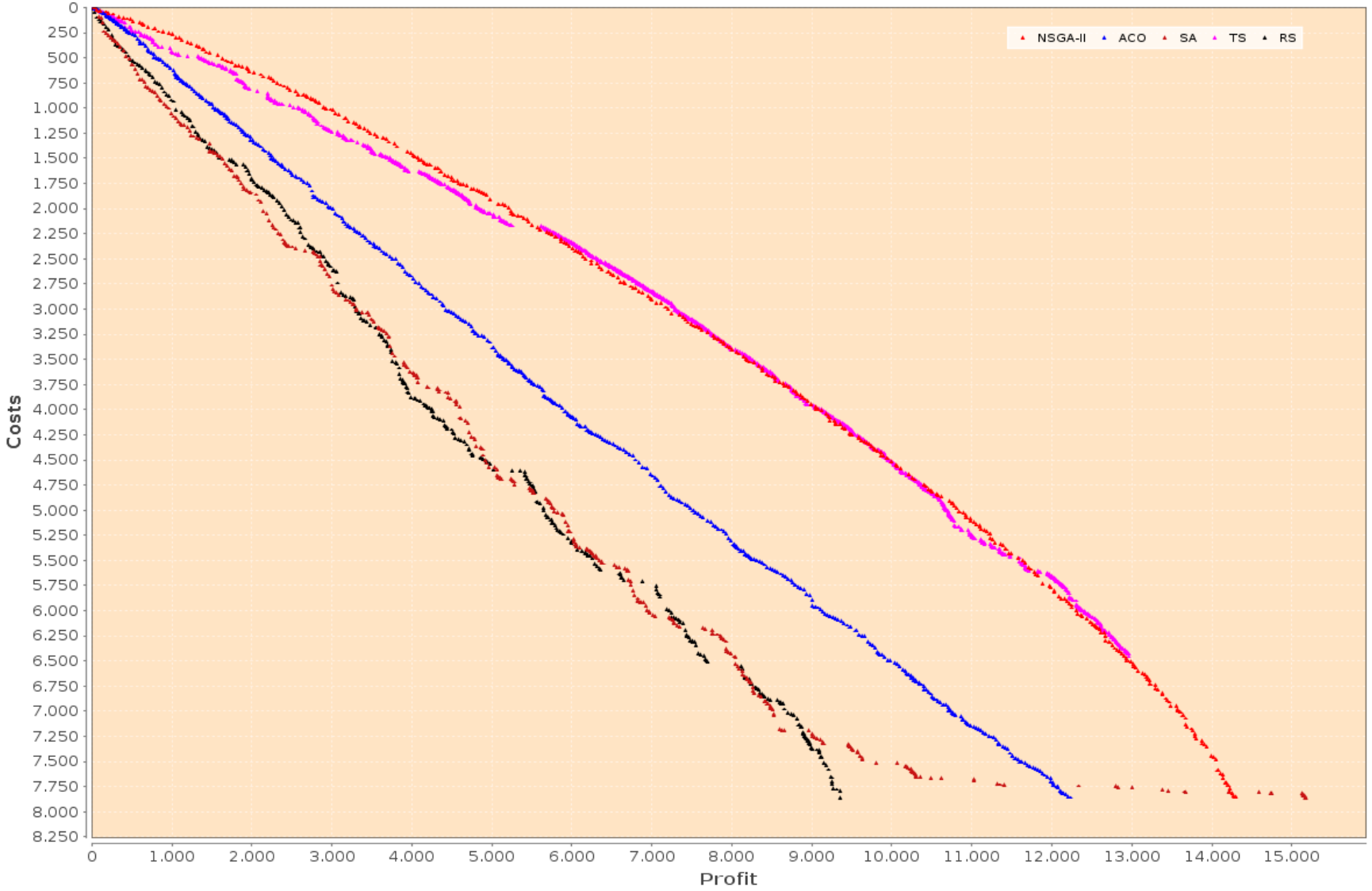
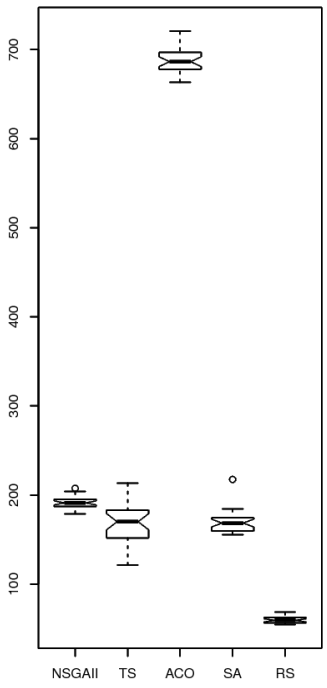
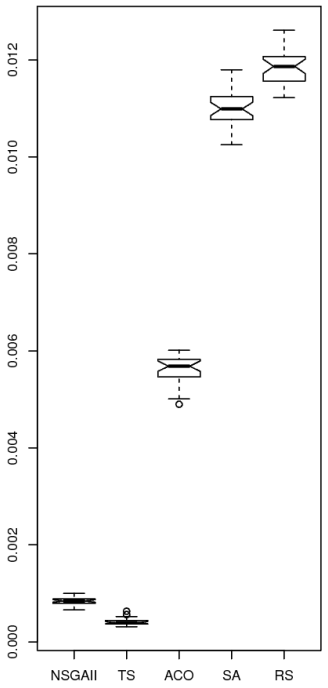


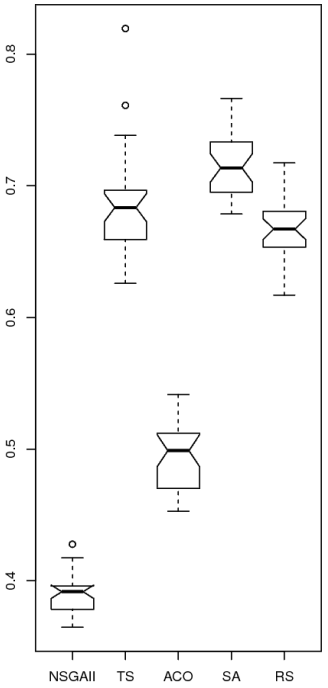
Abbildung 29: Scatterplot für nrp-m1



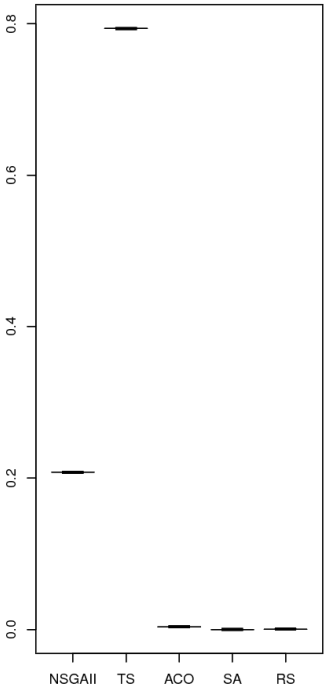
nrp-m1: Time(s)



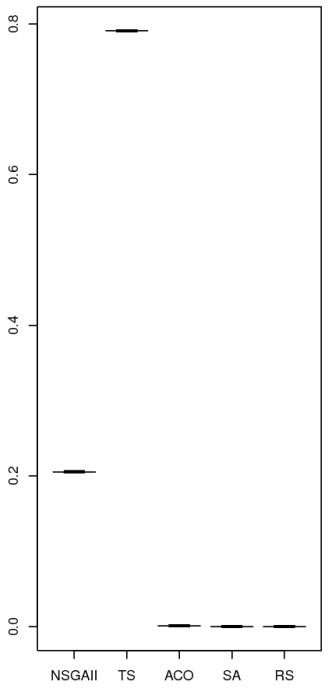
nrp-m1: Conv



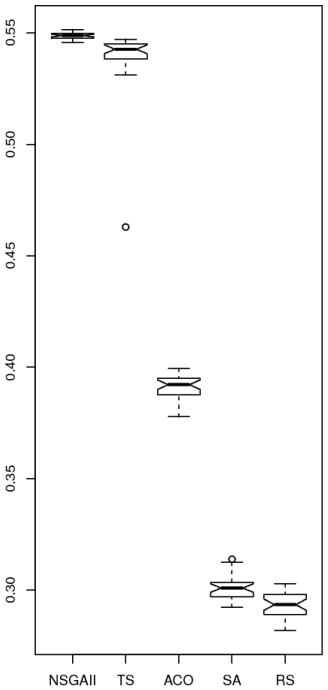
nrp-m1: Diversity



nrp-m1: Contrib



nrp-m1: UContrib



nrp-m1: HVol

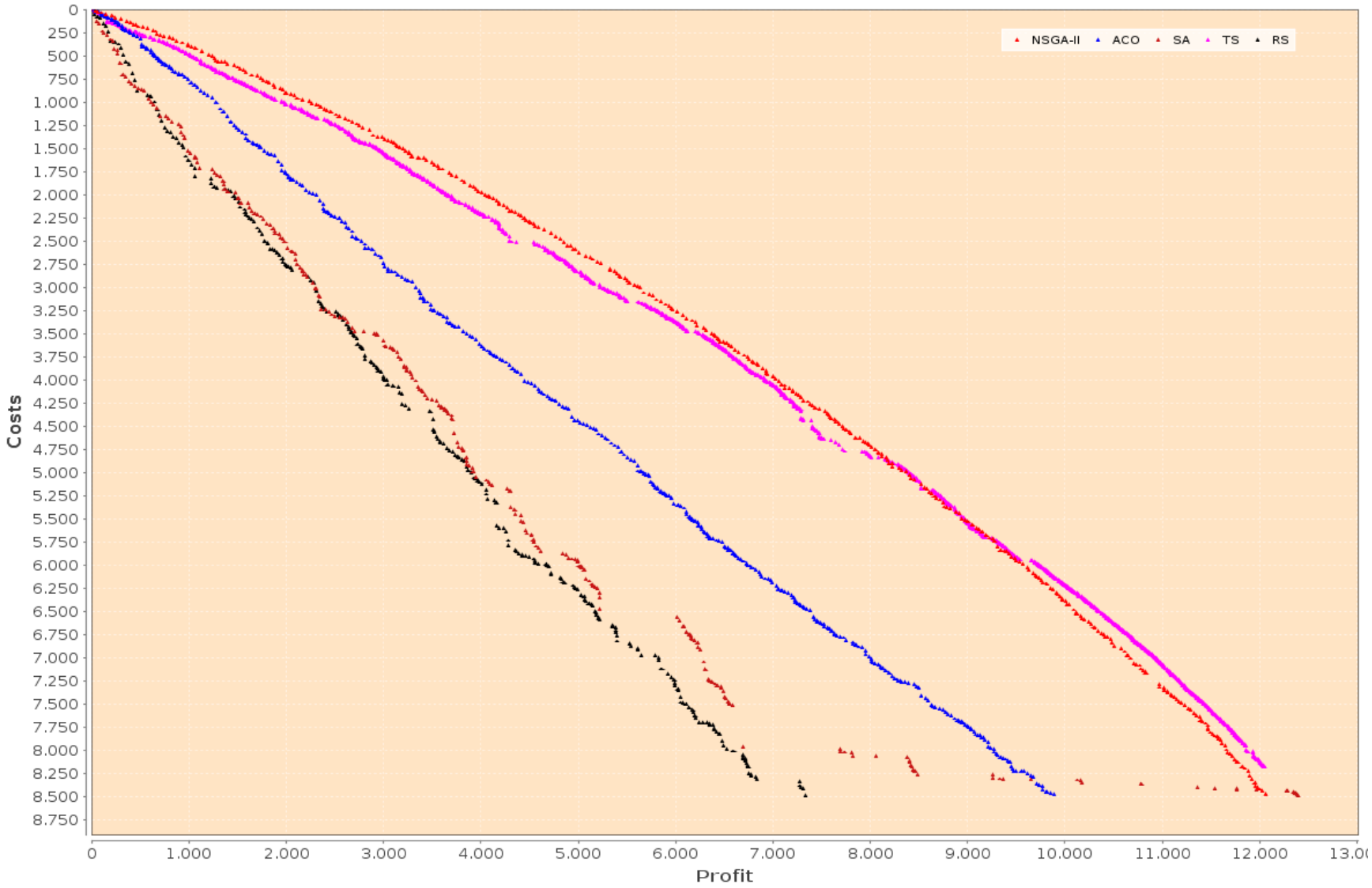
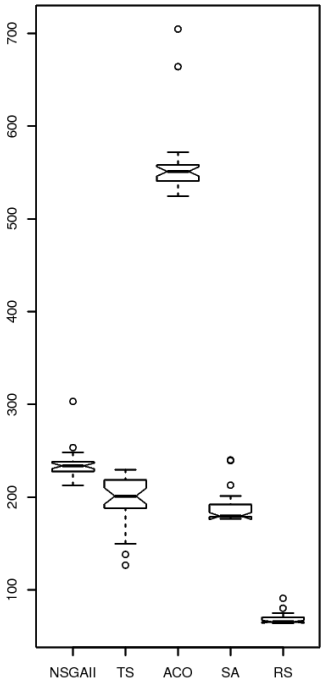
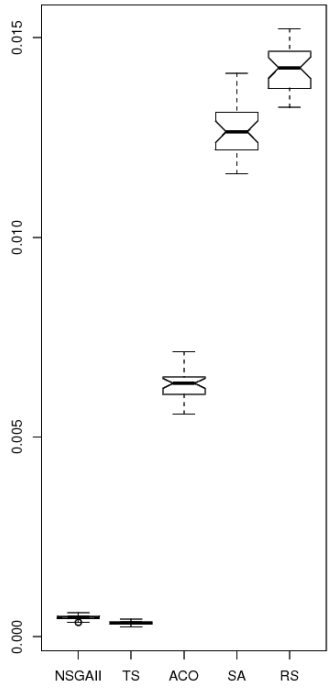


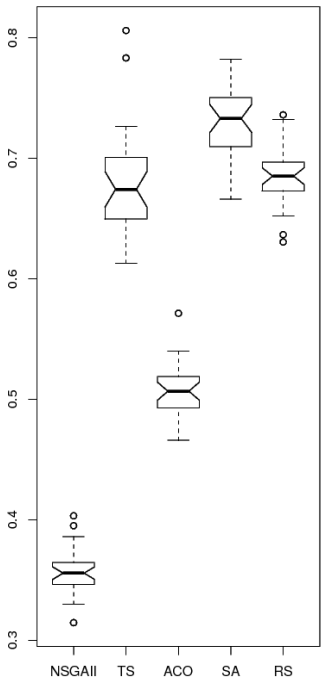
Abbildung 30: Scatterplot für nrp-m2



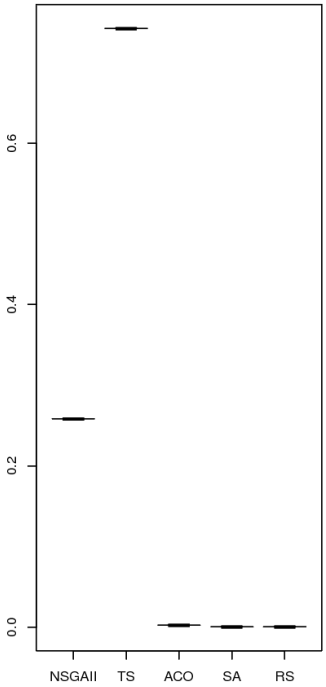
nrp-m2: Time(s)



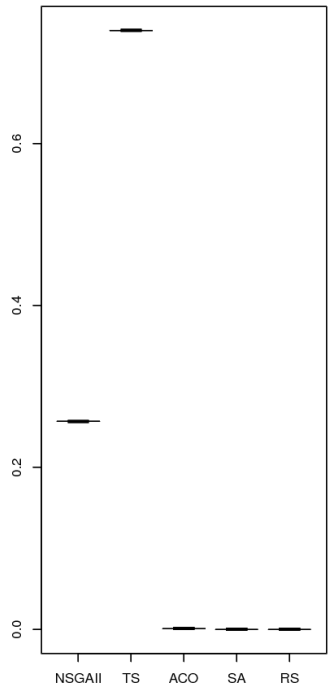
nrp-m2: Conv



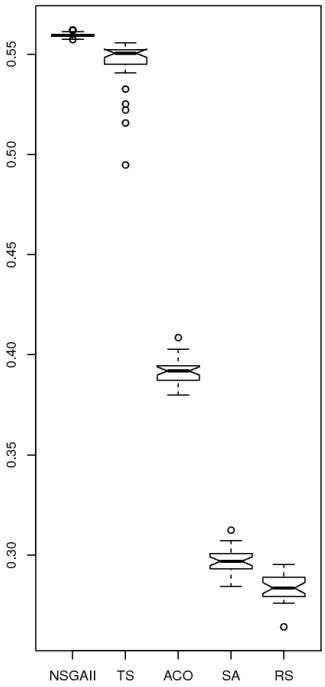
nrp-m2: Diversity



nrp-m2: Contrib



nrp-m2: UContrib



nrp-m2: HVol

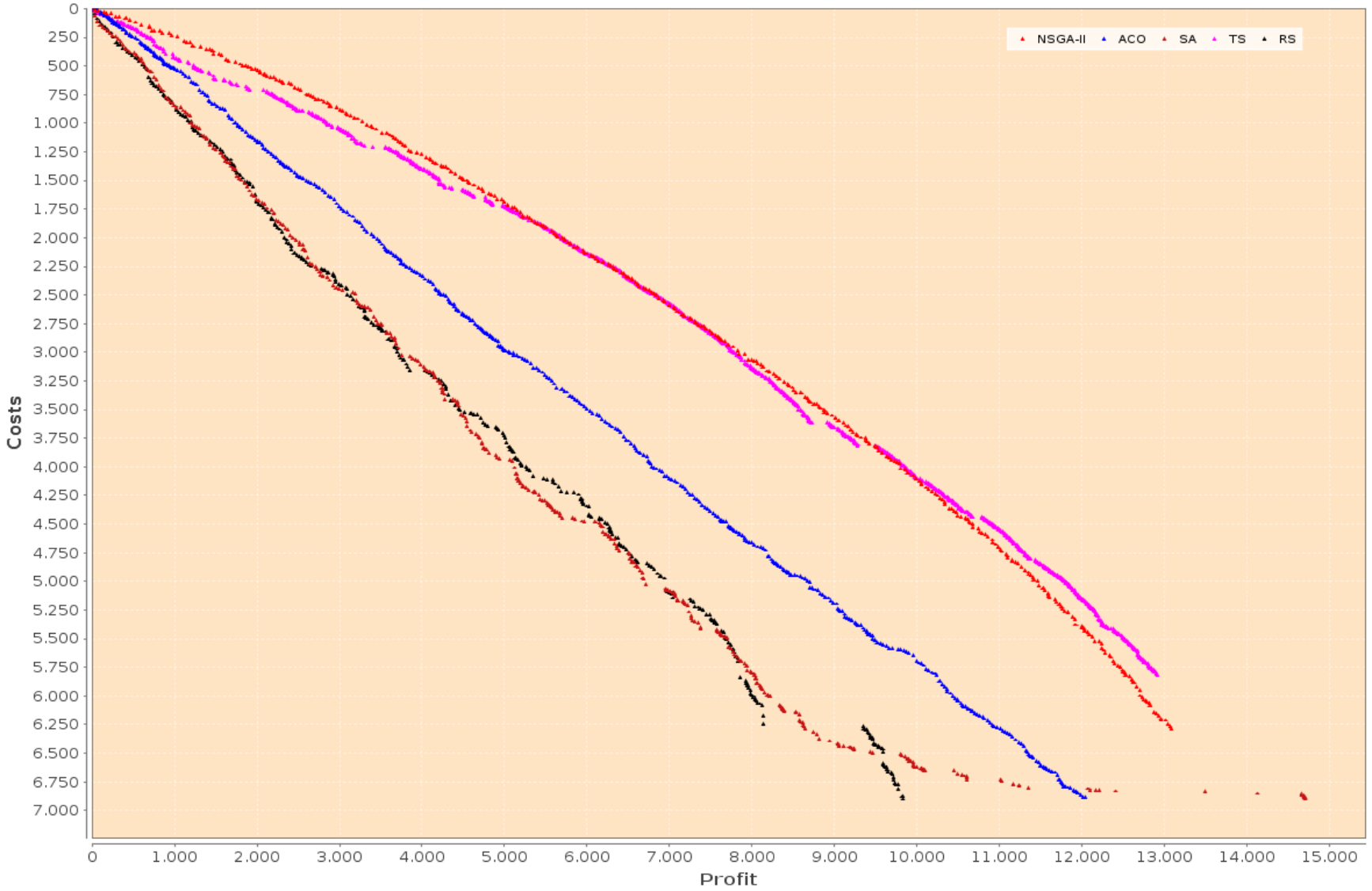
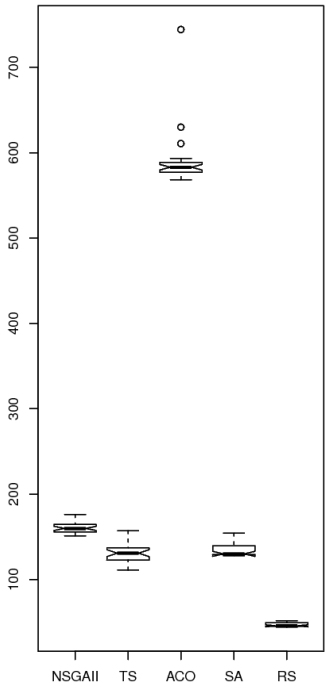
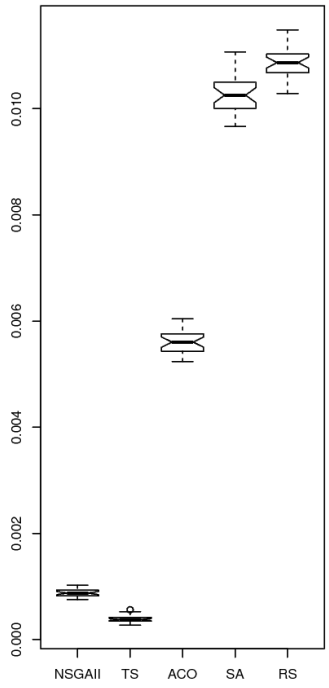


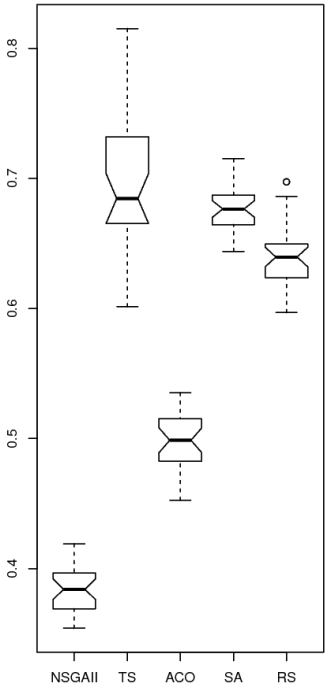
Abbildung 31: Scatterplot für nrp-m3



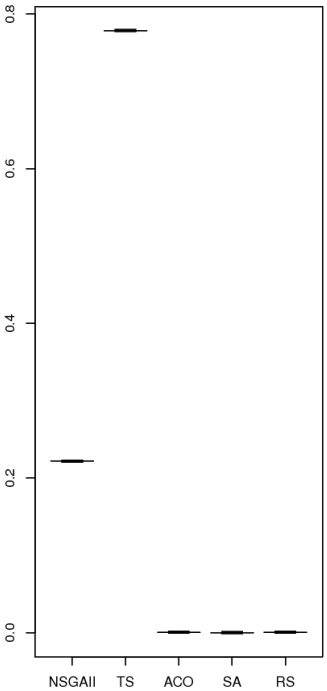
nrp-m3: Time(s)



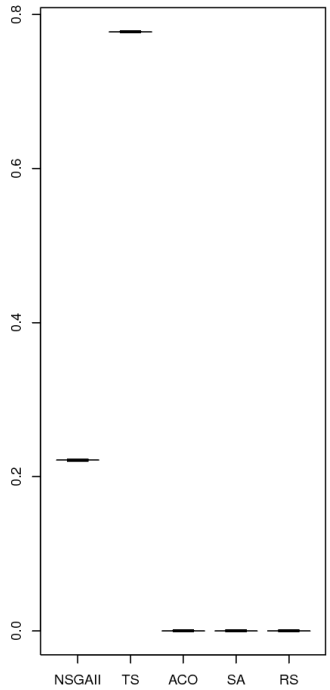
nrp-m3: Conv



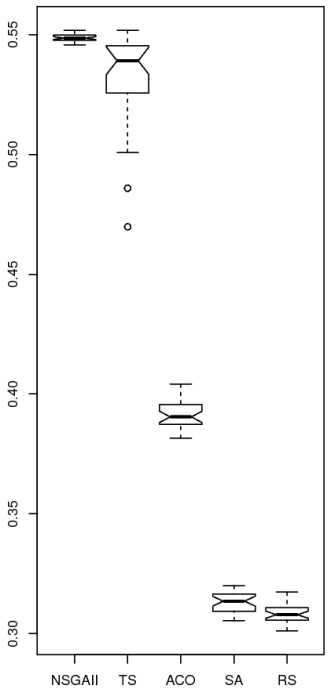
nrp-m3: Diversity



nrp-m3: Contrib



nrp-m3: UContrib



nrp-m3: HVol

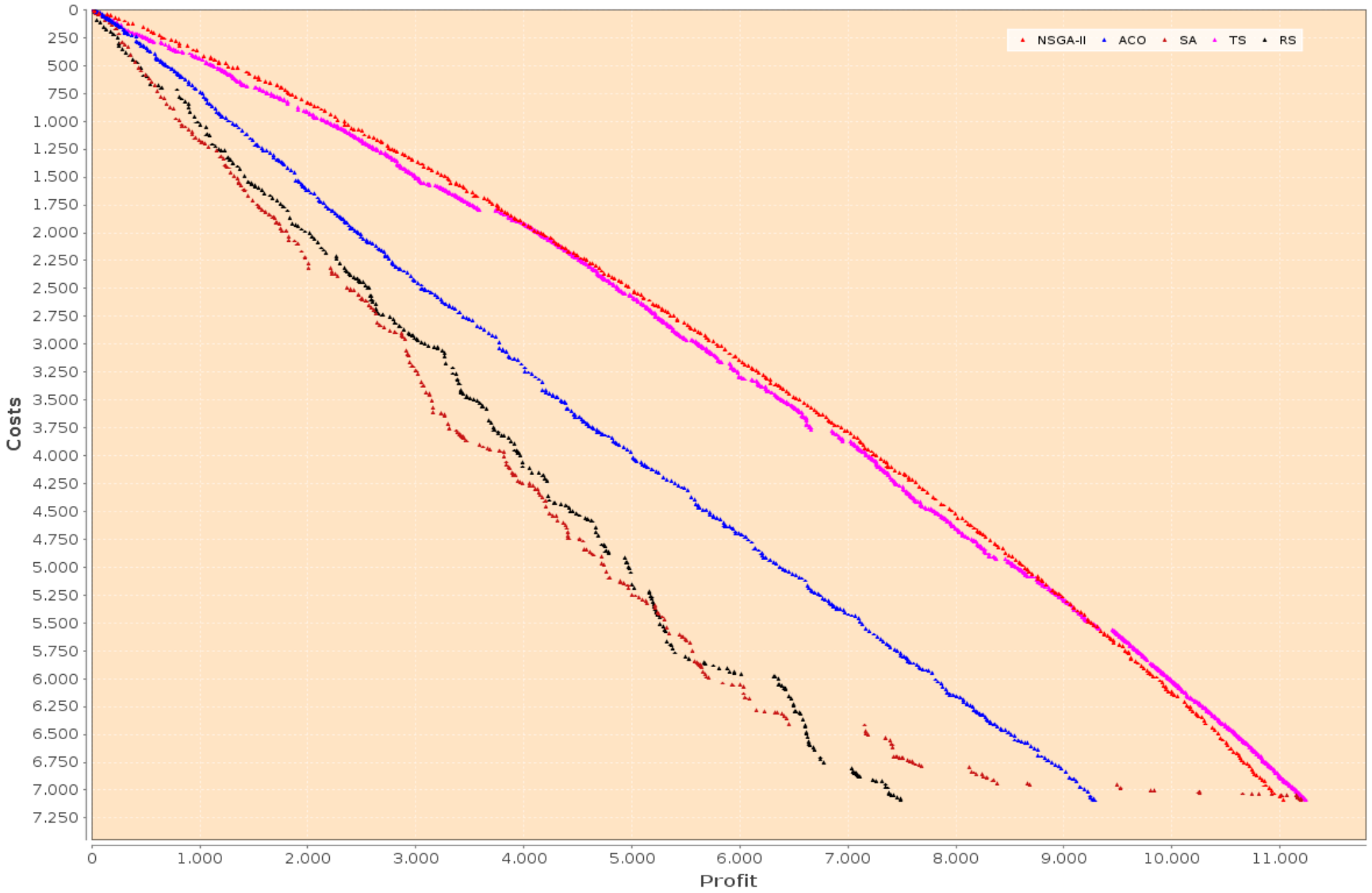
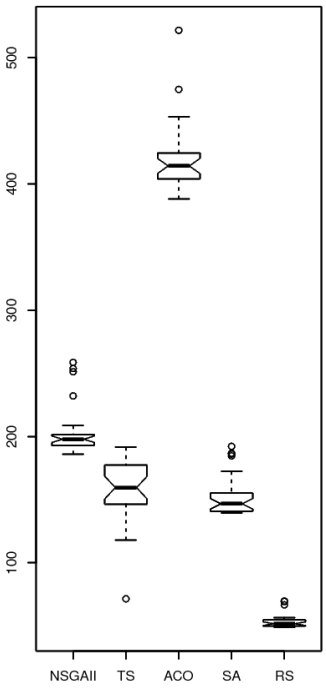
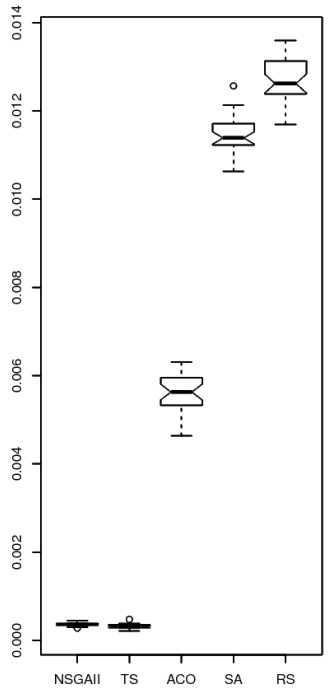


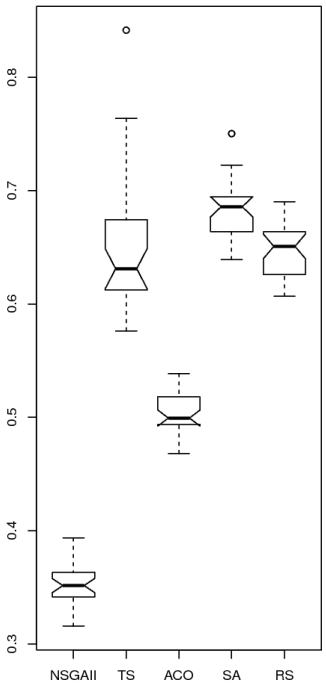
Abbildung 32: Scatterplot für nrp-m4



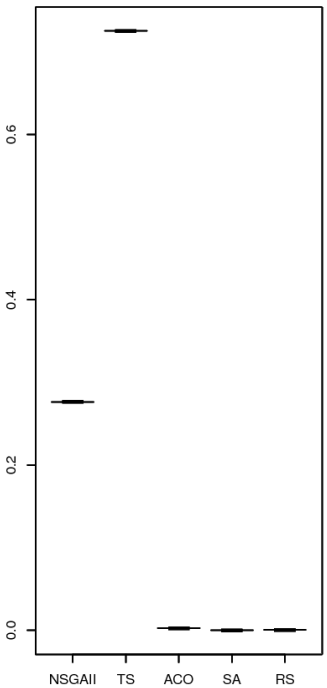
nrp-m4: Time(s)



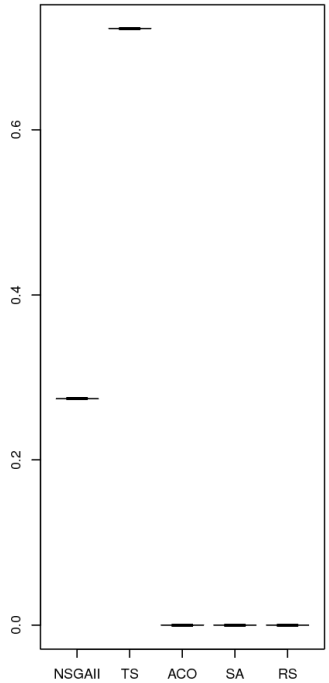
nrp-m4: Conv



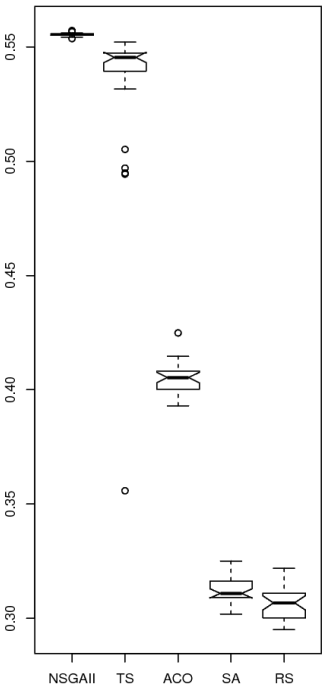
nrp-m4: Diversity



nrp-m4: Contrib



nrp-m4: UContrib



nrp-m4: HVol

7.2.3 Eclipse-Datensatz

Datensatz	Metrik		GA		ACO		TS		SA		RS	
			Mittelwert	Median	Mittelwert	Median	Mittelwert	Median	Mittelwert	Median	Mittelwert	Median
nrp-e1	Qualität	Contrib	2.82e-01	2.82e-01	4.77e-04	4.77e-04	7.19e-01	7.19e-01	4.77e-04	4.77e-04	4.77e-04	4.77e-04
		UContrib	2.81e-01	2.81e-01	0.00e+00	0.00e+00	7.17e-01	7.17e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00
		Conv	6.73e-04	6.87e-04	7.24e-03	7.20e-03	4.45e-04	4.44e-04	1.24e-02	1.25e-02	1.31e-02	1.31e-02
		HVol	5.70e-01	5.71e-01	3.94e-01	3.94e-01	5.62e-01	5.65e-01	3.26e-01	3.27e-01	3.20e-01	3.19e-01
	Diversity		4.60e-01	4.52e-01	5.20e-01	5.20e-01	7.01e-01	6.91e-01	7.20e-01	7.25e-01	7.04e-01	7.04e-01
	Time		1.16e+02	1.15e+02	2.13e+02	2.11e+02	1.21e+02	1.23e+02	1.06e+02	1.02e+02	4.36e+01	4.19e+01
nrp-e2	Qualität	Contrib	2.85e-01	2.85e-01	1.48e-03	1.48e-03	7.17e-01	7.17e-01	4.92e-04	4.92e-04	4.92e-04	4.92e-04
		UContrib	2.83e-01	2.83e-01	0.00e+00	0.00e+00	7.15e-01	7.15e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00
		Conv	6.10e-04	5.90e-04	7.56e-03	7.55e-03	4.28e-04	4.30e-04	1.39e-02	1.39e-02	1.58e-02	1.58e-02
		HVol	5.77e-01	5.77e-01	3.96e-01	3.97e-01	5.64e-01	5.69e-01	3.09e-01	3.09e-01	2.96e-01	2.96e-01
	Diversity		4.56e-01	4.61e-01	5.11e-01	5.07e-01	7.15e-01	7.04e-01	7.06e-01	7.01e-01	6.78e-01	6.75e-01
	Time		1.64e+02	1.58e+02	2.57e+02	2.54e+02	1.65e+02	1.66e+02	1.84e+02	1.75e+02	6.63e+01	6.35e+01
nrp-e3	Qualität	Contrib	2.75e-01	2.75e-01	1.38e-03	1.38e-03	7.28e-01	7.28e-01	0.00e+00	0.00e+00	4.59e-04	4.59e-04
		UContib	2.72e-01	2.72e-01	0.00e+00	0.00e+00	7.23e-01	7.23e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00
		Conv	4.27e-04	4.30e-04	6.61e-03	6.52e-03	4.05e-04	3.90e-04	1.21e-02	1.21e-02	1.32e-02	1.33e-02
		HVol	5.80e-01	5.80e-01	4.17e-01	4.17e-01	5.63e-01	5.66e-01	3.37e-01	3.37e-01	3.30e-01	3.29e-01
	Diversity		4.20e-01	4.17e-01	4.98e-01	5.00e-01	7.18e-01	7.12e-01	6.59e-01	6.59e-01	6.34e-01	6.31e-01
	Time		1.00e+02	9.86e+01	1.45e+02	1.35e+02	9.48e+01	9.53e+01	9.80e+01	9.69e+01	3.33e+01	3.29e+01
nrp-e4	Qualität	Contrib	5.08e-01	5.08e-01	1.61e-03	1.61e-03	4.98e-01	4.98e-01	0.00e+00	0.00e+00	5.38e-04	5.38e-04
		UContrib	5.01e-01	5.01e-01	0.00e+00	0.00e+00	4.92e-01	4.92e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00
		Conv	2.89e-04	2.95e-04	7.50e-03	7.46e-03	3.54e-04	3.37e-04	1.32e-02	1.33e-02	1.51e-02	1.51e-02
		HVol	5.85e-01	5.85e-01	4.08e-01	4.10e-01	5.58e-01	5.72e-01	3.32e-01	3.32e-01	3.21e-01	3.20e-01
	Diversity		4.03e-01	4.04e-01	5.19e-01	5.26e-01	6.85e-01	6.80e-01	6.50e-01	6.49e-01	6.39e-01	6.37e-01
	Time		1.20e+02	1.20e+02	1.24e+02	1.25e+02	1.04e+02	1.10e+02	1.07e+02	1.05e+02	3.68e+01	3.64e+01

Tabelle 10: Leistung der Algorithmen (Mittelwert/Median) für Eclipse-Datensatz

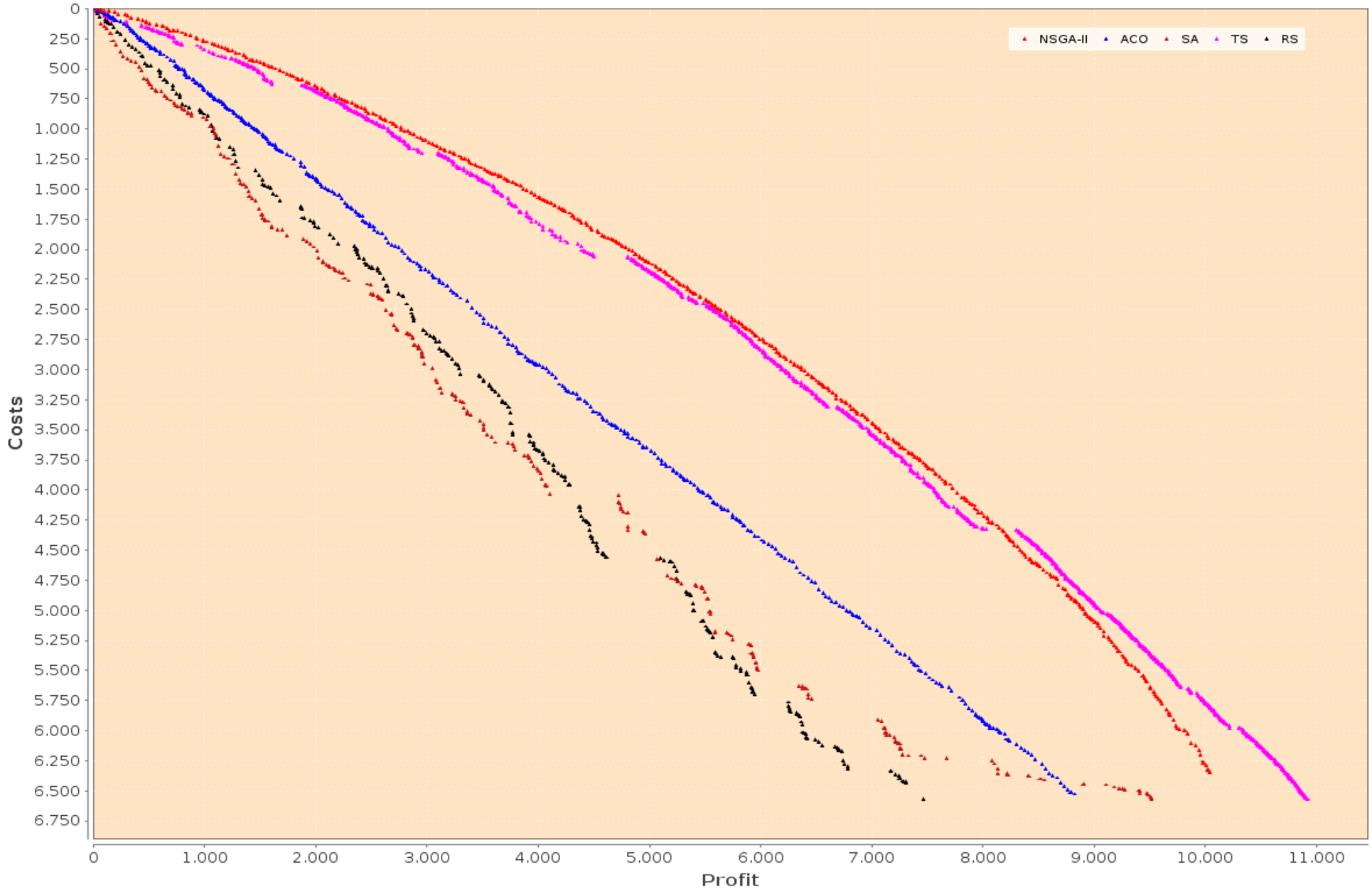
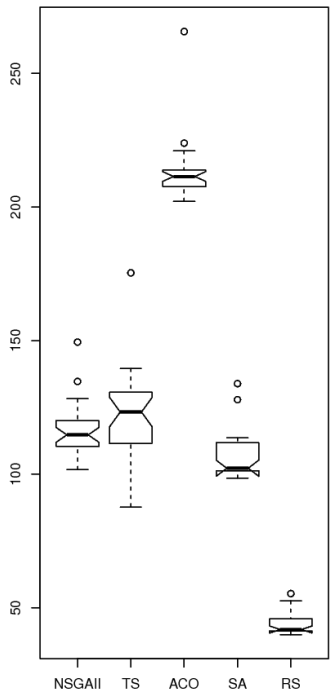
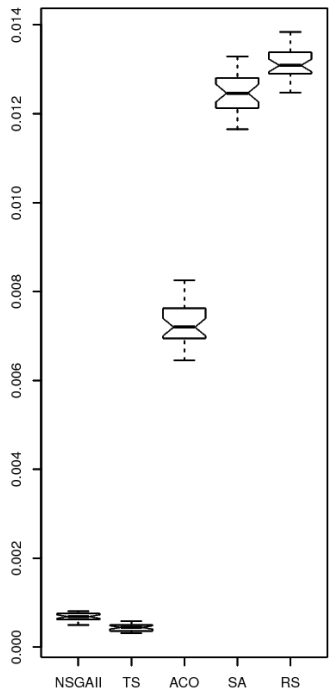


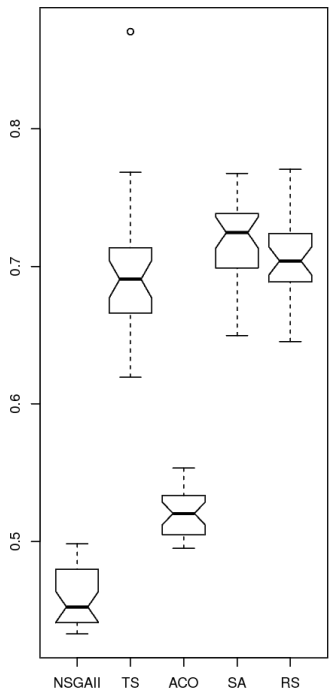
Abbildung 33: Scatterplot für nrp-e1



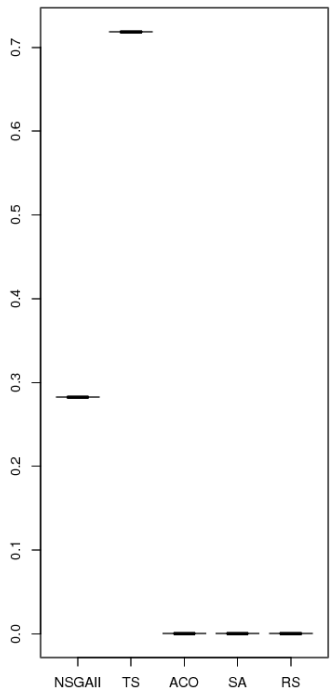
nrp-e1: Time(s)



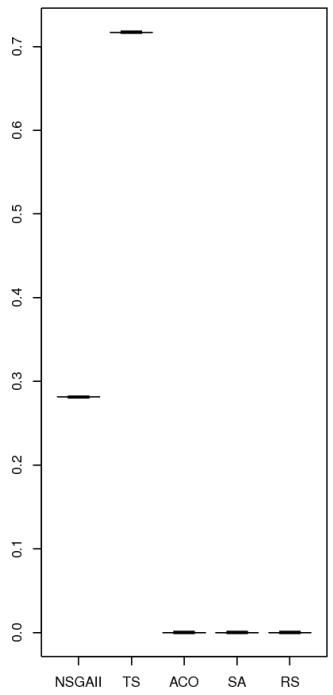
nrp-e1: Conv



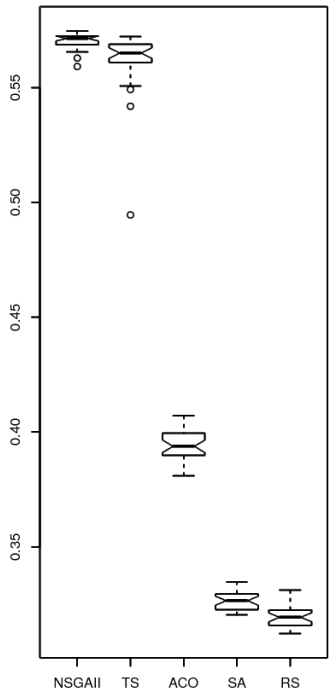
nrp-e1: Diversity



nrp-e1: Contrib



nrp-e1: UContrib



nrp-e1: HVol

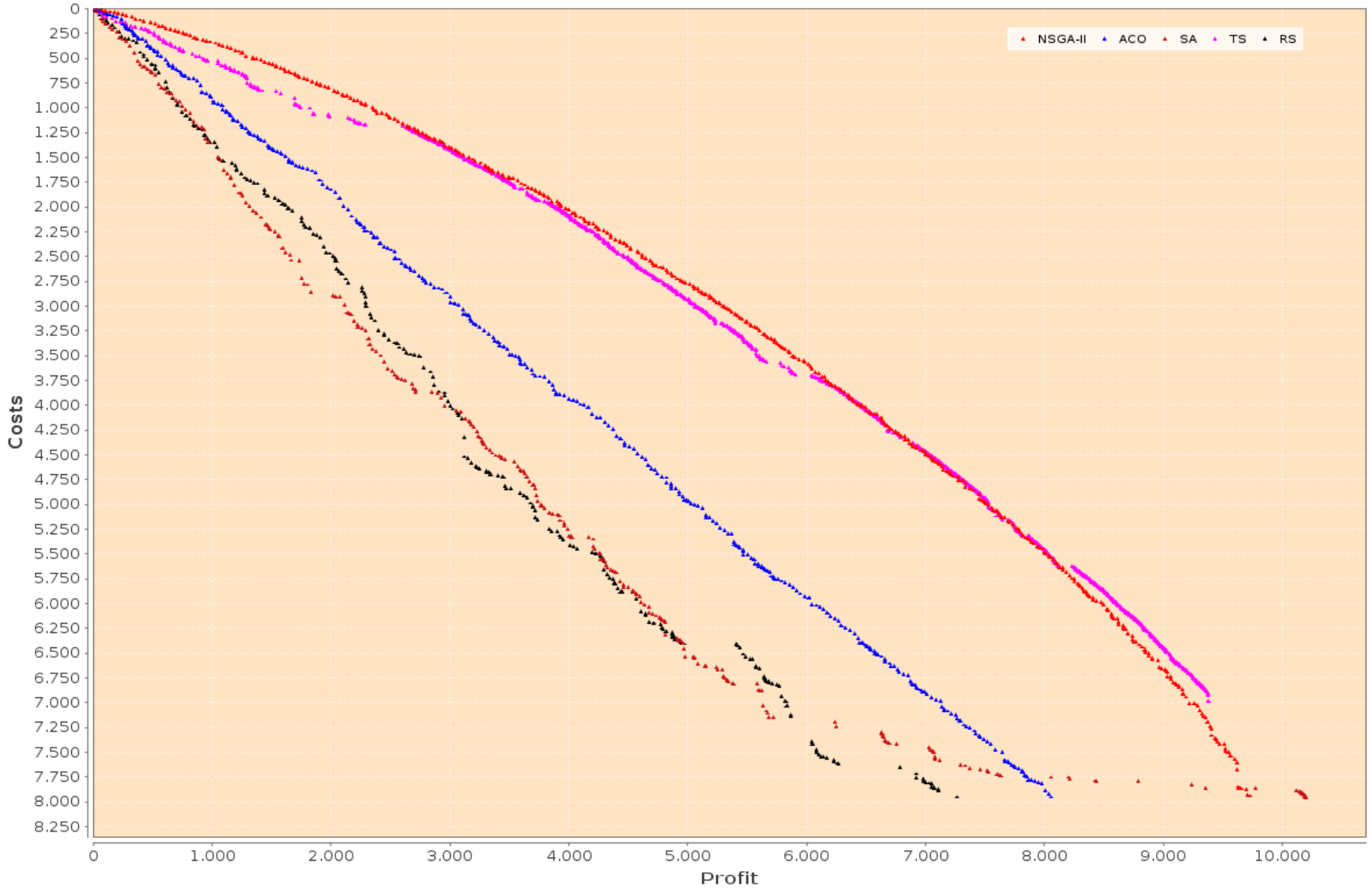
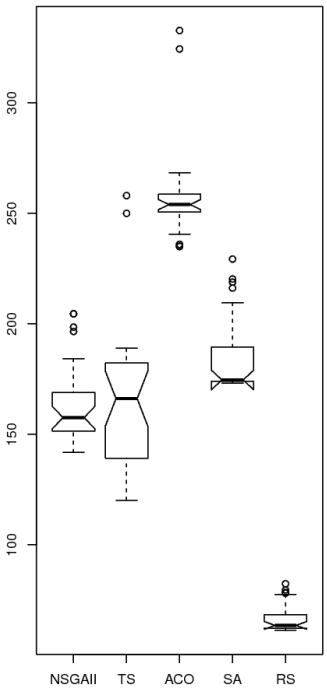
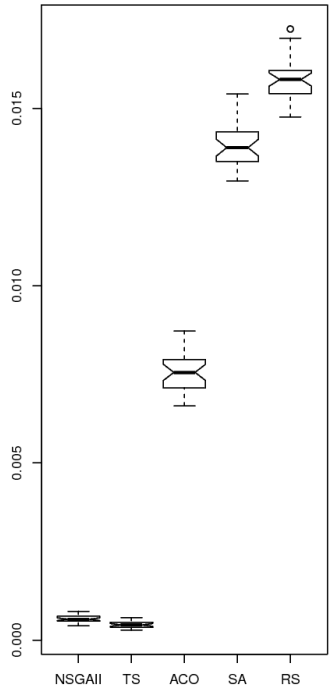


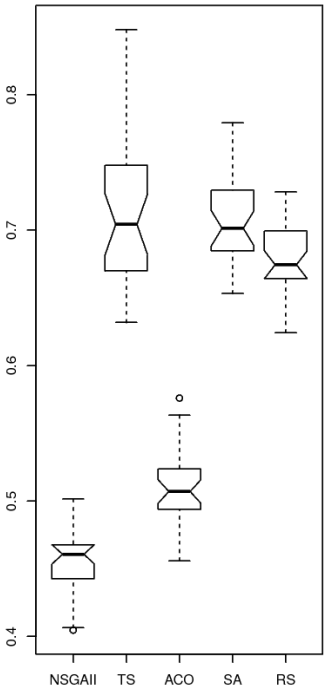
Abbildung 34: Scatterplot für nrp-e2



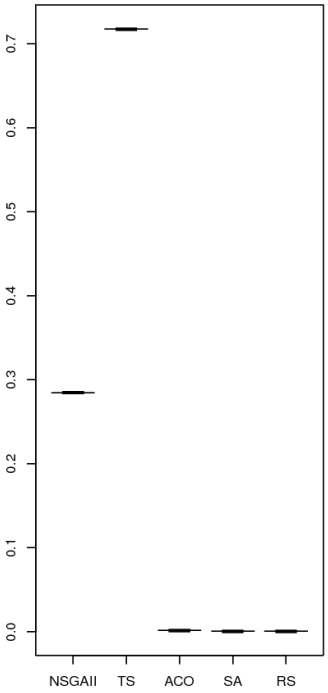
nrp-e2: Time(s)



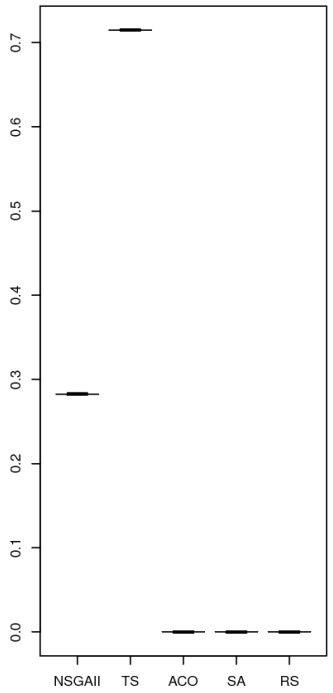
nrp-e2: Conv



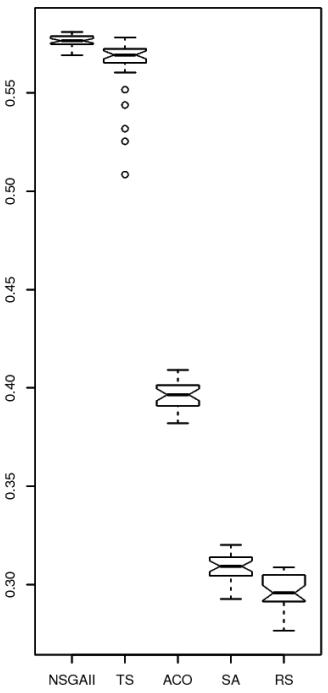
nrp-e2: Diversity



nrp-e2: Contrib



nrp-e2: UContrib



nrp-e2: HVol

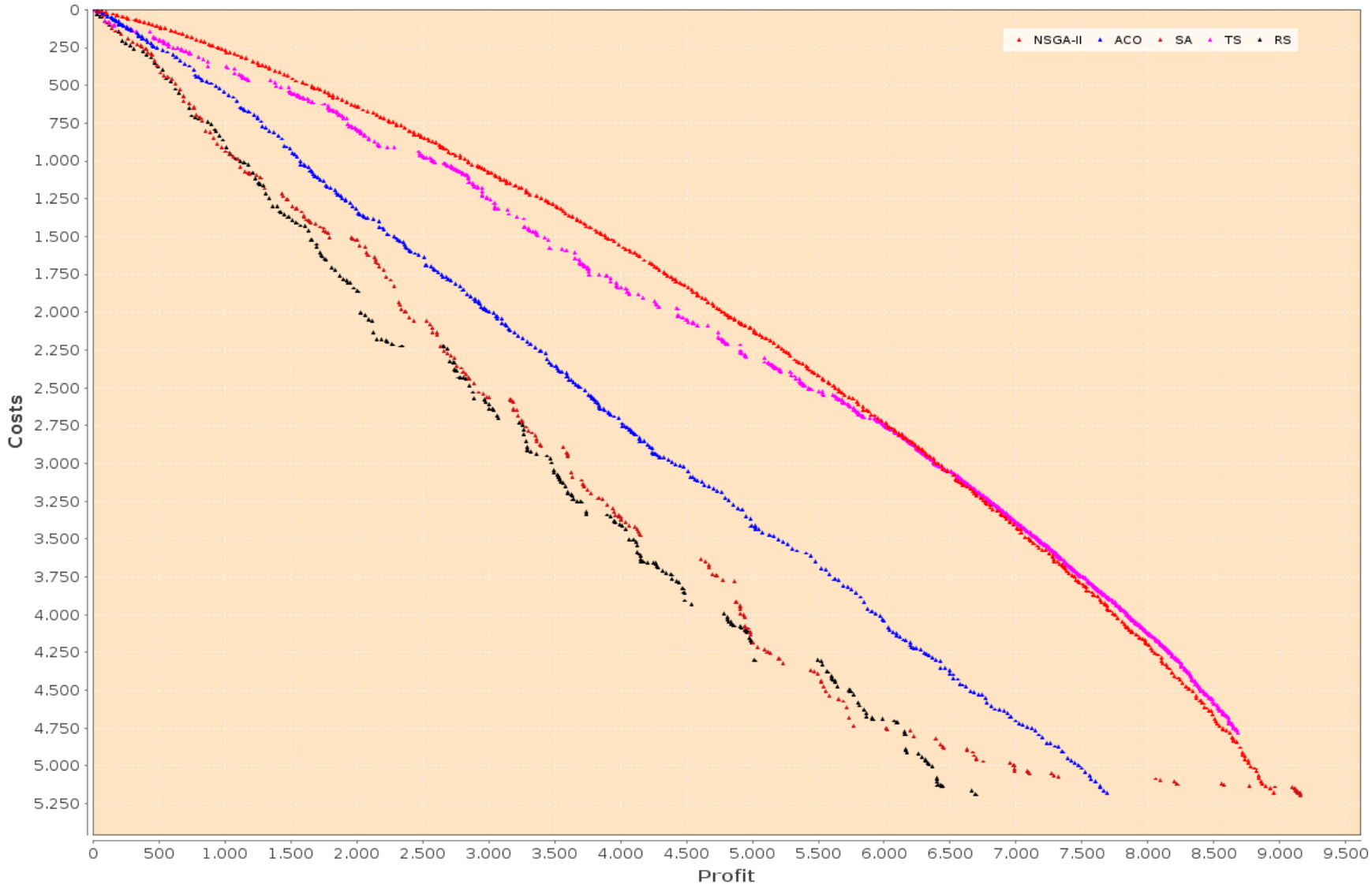
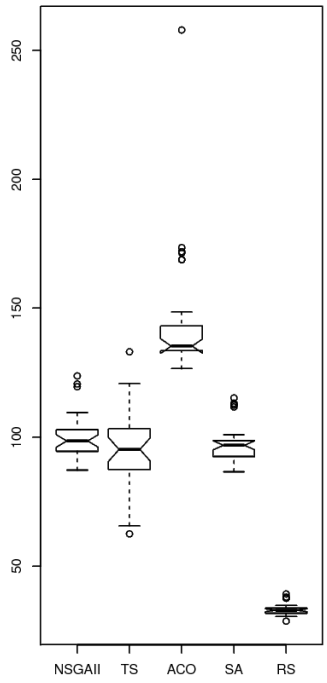
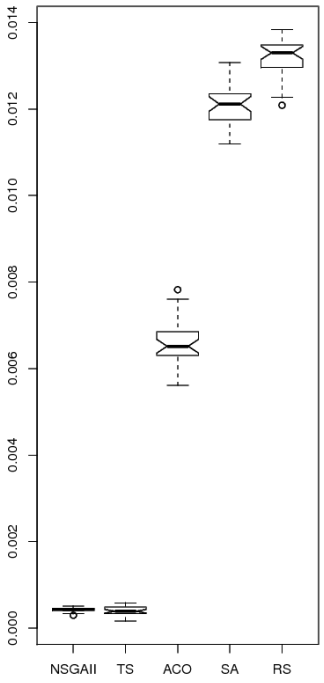


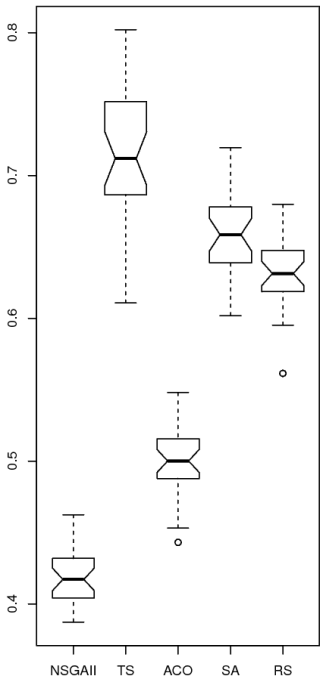
Abbildung 35: Scatterplot für nrp-e3



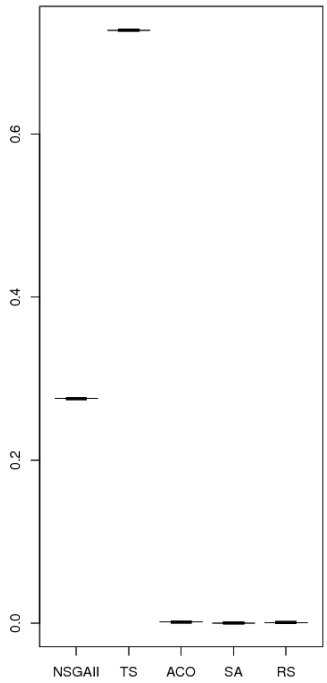
nrp-e3: Time(s)



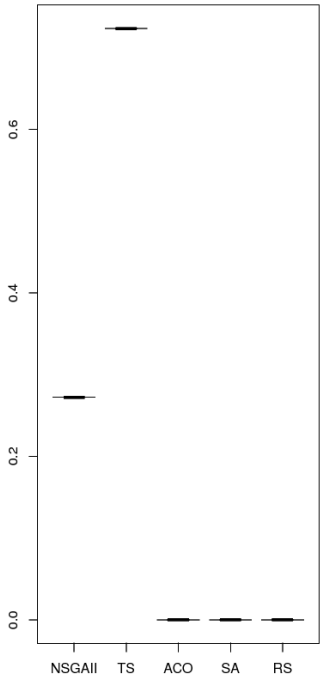
nrp-e3: Conv



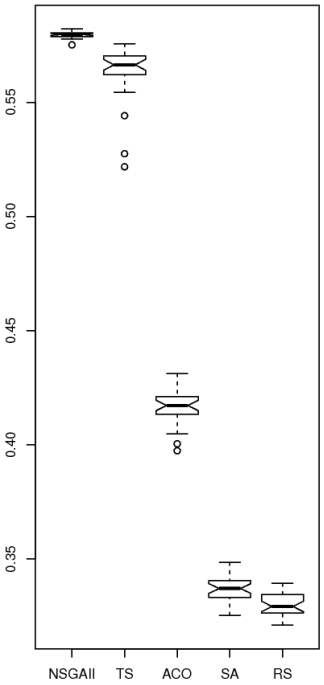
nrp-e3: Diversity



nrp-e3: Contrib



nrp-e3: UContrib



nrp-e3: HVol

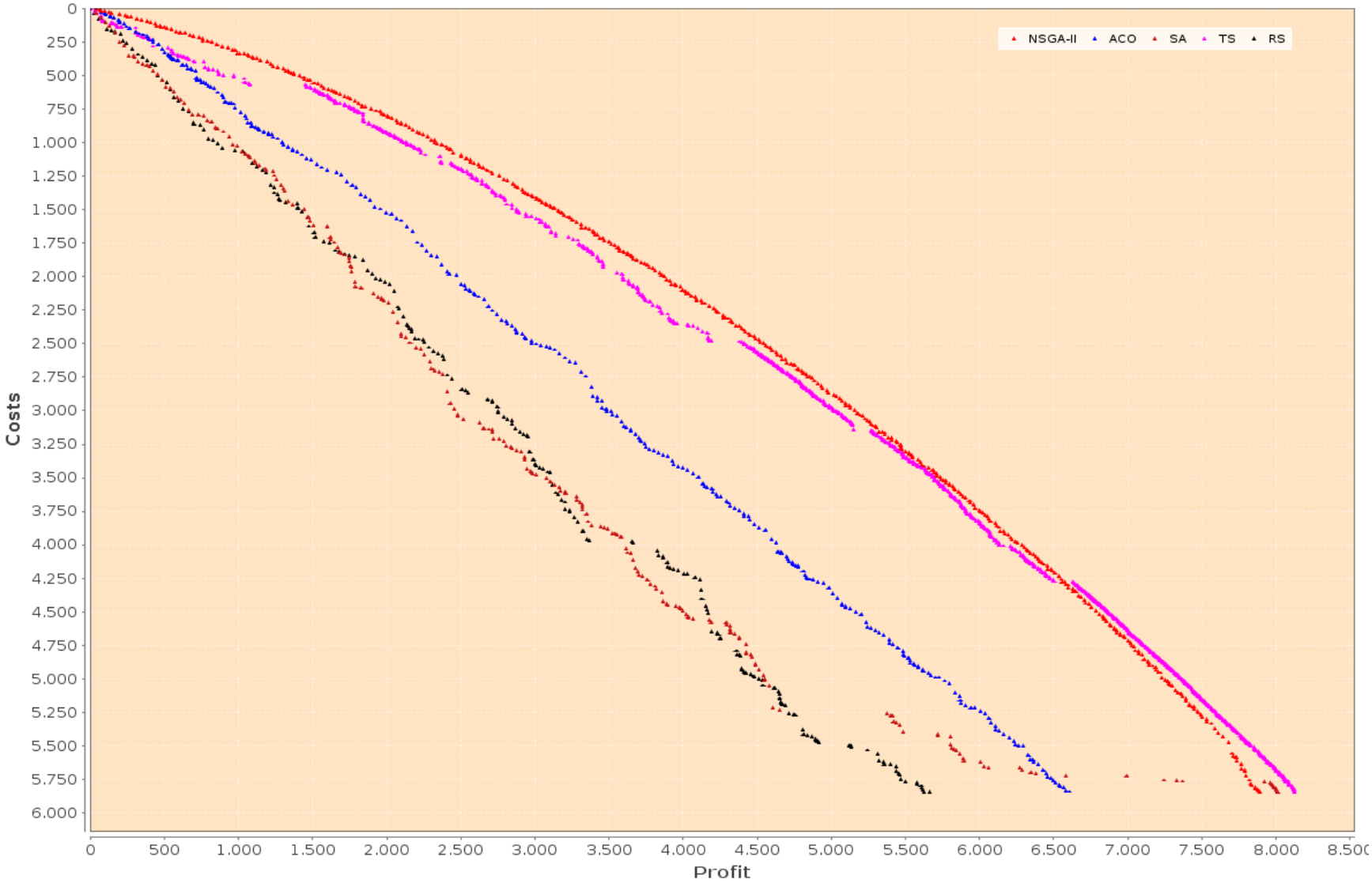
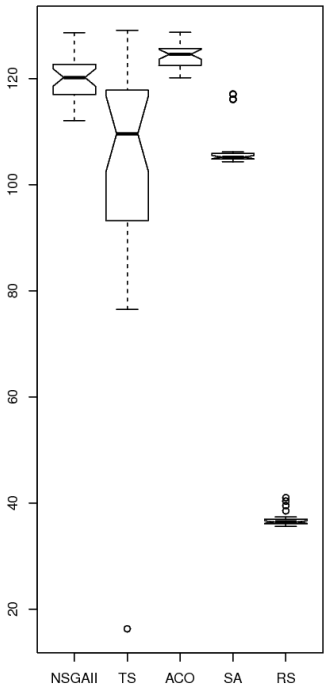
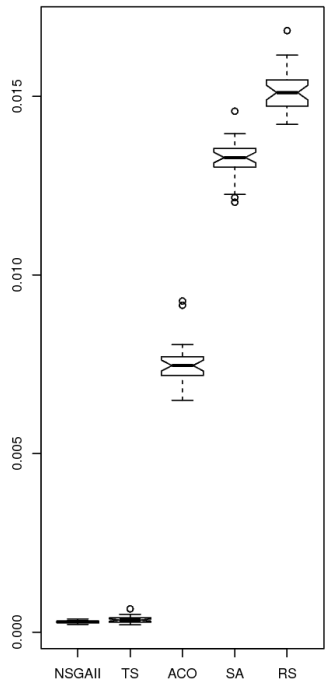


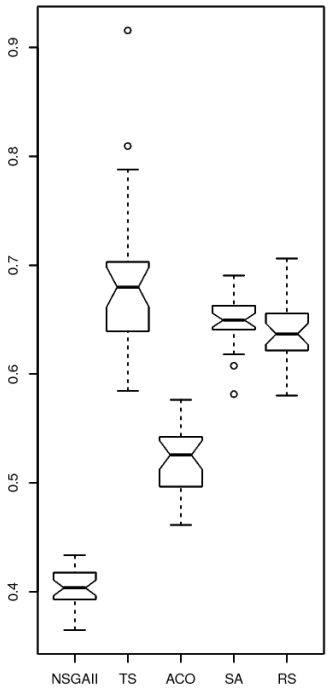
Abbildung 36: Scatterplot für nrp-e4



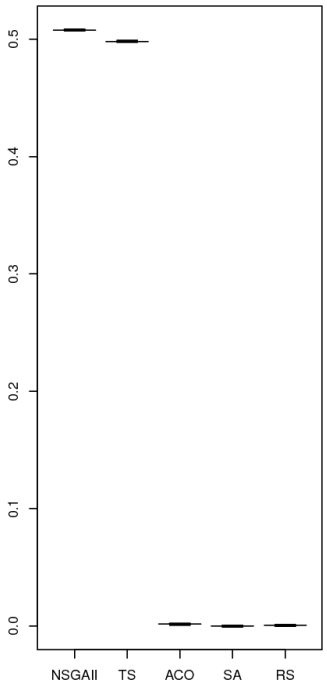
nrp-e4: Time(s)



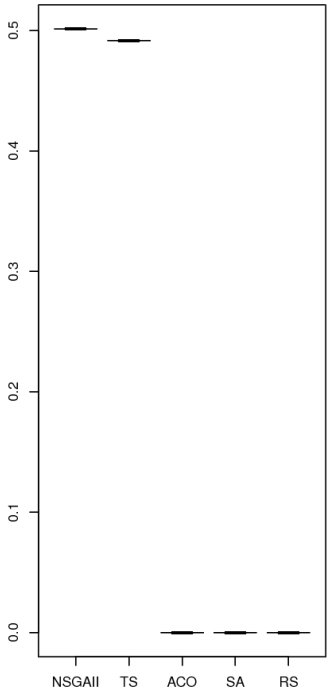
nrp-e4: Conv



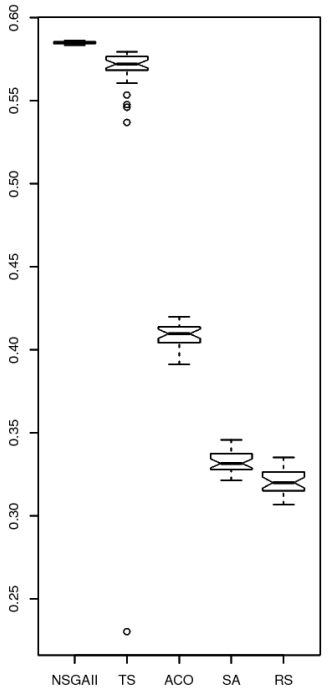
nrp-e4: Diversity



nrp-e4: Contrib



nrp-e4: UContrib



nrp-e4: HVol

7.2.4 Gnome-Datensatz

Datensatz	Metrik		GA		ACO		TS		SA		RS	
			Mittelwert	Median	Mittelwert	Median	Mittelwert	Median	Mittelwert	Median	Mittelwert	Median
nrp-g1	Qualität	Contrib	1.79e-01	1.79e-01	1.39e-03	1.39e-03	8.34e-01	8.34e-01	0.00e+00	0.00e+00	3.47e-04	3.47e-04
		UContrib	1.66e-01	1.66e-01	0.00e+00	0.00e+00	8.21e-01	8.21e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00
		Conv	3.90e-04	3.99e-04	5.18e-03	5.17e-03	1.65e-04	1.56e-04	9.53e-03	9.57e-03	1.04e-02	1.03e-02
		HVol	5.61e-01	5.61e-01	4.28e-01	4.30e-01	5.26e-01	5.48e-01	3.62e-01	3.63e-01	3.57e-01	3.57e-01
	Diversity		3.98e-01	3.99e-01	5.24e-01	5.27e-01	6.79e-01	6.69e-01	6.26e-01	6.29e-01	6.05e-01	6.03e-01
	Time		8.45e+01	8.52e+01	1.23e+02	1.22e+02	9.82e+01	1.03e+02	8.09e+01	8.25e+01	2.80e+01	2.86e+01
nrp-g2	Qualität	Contrib	3.80e-01	3.80e-01	8.67e-04	8.67e-04	6.44e-01	6.44e-01	0.00e+00	0.00e+00	4.33e-04	4.33e-04
		UContrib	3.56e-01	3.56e-01	0.00e+00	0.00e+00	6.20e-01	6.20e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00
		Conv	1.20e-04	1.17e-04	6.02e-03	6.00e-03	1.77e-04	1.80e-04	1.20e-02	1.21e-02	1.39e-02	1.38e-02
		HVol	5.76e-01	5.76e-01	4.33e-01	4.31e-01	5.34e-01	5.68e-01	3.55e-01	3.54e-01	3.45e-01	3.46e-01
	Diversity		3.69e-01	3.69e-01	5.28e-01	5.25e-01	6.21e-01	5.79e-01	6.74e-01	6.76e-01	6.51e-01	6.46e-01
	Time		9.34e+01	9.25e+01	7.71e+01	7.64e+01	7.68e+01	8.85e+01	7.97e+01	8.02e+01	2.84e+01	2.87e+01
nrp-g3	Qualität	Contrib	4.63e-01	4.63e-01	1.41e-03	1.41e-03	5.47e-01	5.47e-01	0.00e+00	0.00e+00	4.71e-04	4.71e-04
		UContib	4.53e-01	4.53e-01	0.00e+00	0.00e+00	5.36e-01	5.36e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00
		Conv	2.43e-04	2.37e-04	4.93e-03	4.92e-03	1.95e-04	1.82e-04	1.00e-02	1.00e-02	1.11e-02	1.11e-02
		HVol	5.75e-01	5.75e-01	4.44e-01	4.45e-01	5.51e-01	5.69e-01	3.66e-01	3.66e-01	3.58e-01	3.58e-01
	Diversity		3.89e-01	3.93e-01	5.07e-01	5.00e-01	6.02e-01	5.68e-01	6.31e-01	6.32e-01	6.08e-01	6.12e-01
	Time		7.99e+01	7.98e+01	1.09e+02	1.08e+02	7.49e+01	7.64e+01	7.27e+01	7.22e+01	2.54e+01	2.57e+01
nrp-g4	Qualität	Contrib	7.10e-01	7.10e-01	1.56e-03	1.56e-03	2.99e-01	2.99e-01	0.00e+00	0.00e+00	5.19e-04	5.19e-04
		UContrib	7.01e-01	7.01e-01	0.00e+00	0.00e+00	2.89e-01	2.89e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00
		Conv	8.31e-05	7.44e-05_	6.41e-03	6.43e-03	2.09e-04	2.00e-04	1.15e-02	1.16e-02	1.32e-02	1.32e-02
		HVol	5.77e-01	5.77e-01	4.30e-01	4.30e-01	5.61e-01	5.67e-01	3.65e-01	3.64e-01	3.55e-01	3.55e-01
	Diversity		3.73e-01	3.73e-01	5.26e-01	5.27e-01	6.25e-01	6.32e-01	6.30e-01	6.30e-01	6.14e-01	6.10e-01
	Time		7.85e+01	7.80e+01	5.48e+01	5.48e+01	7.16e+01	7.54e+01	6.31e+01	6.29e+01	2.17e+01	2.18e+01

Tabelle 11: Leistung der Algorithmen (Mittelwert/Median) für Gnome-Datensatz

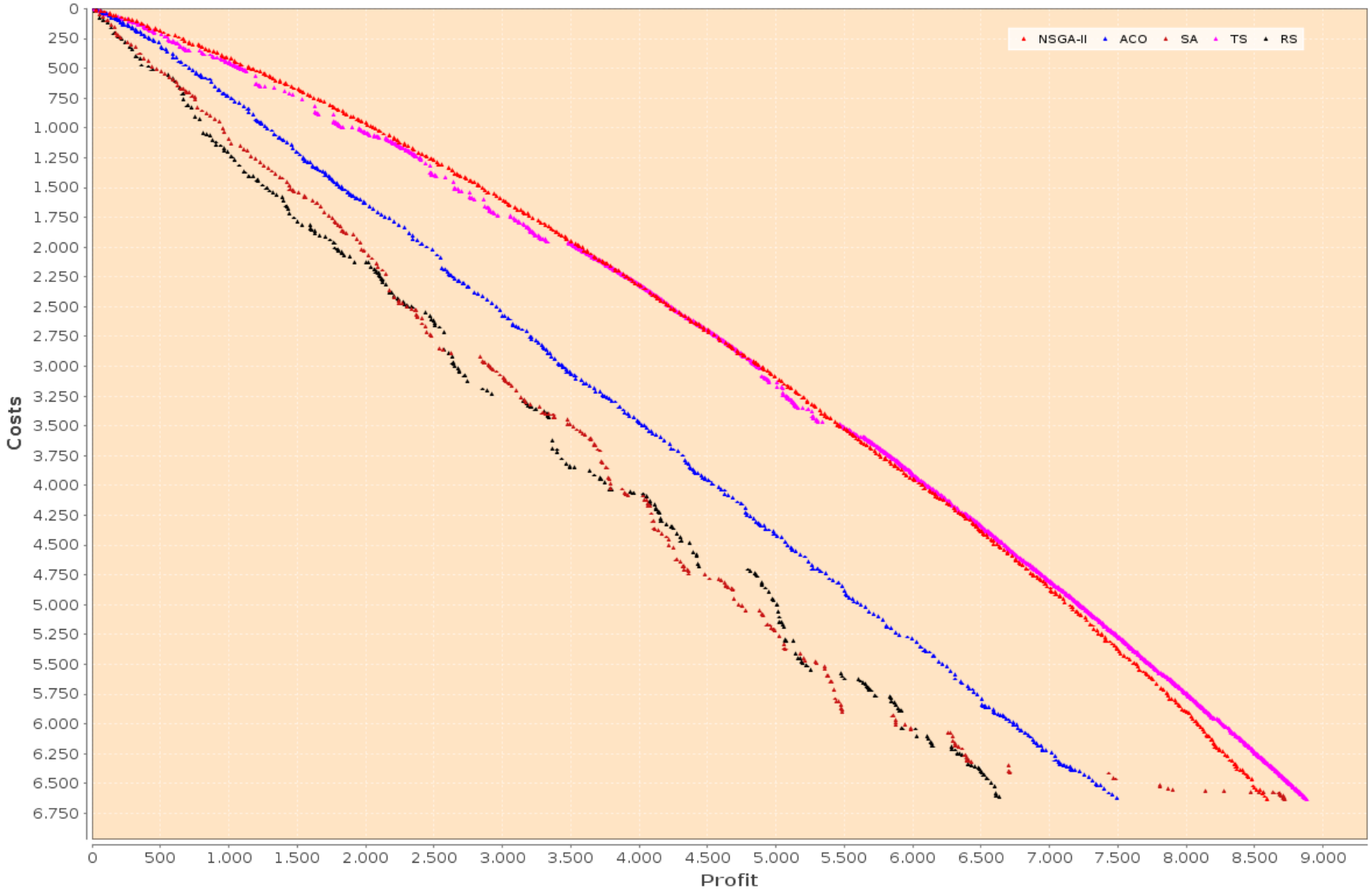
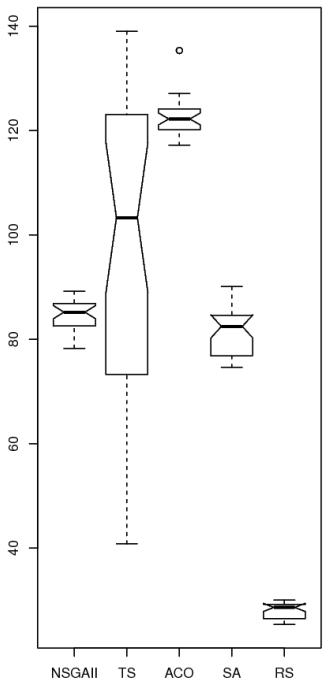
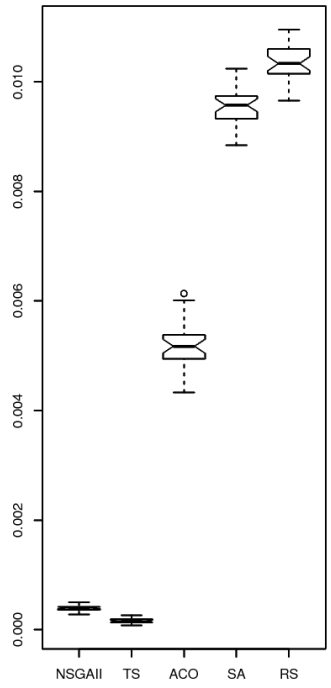


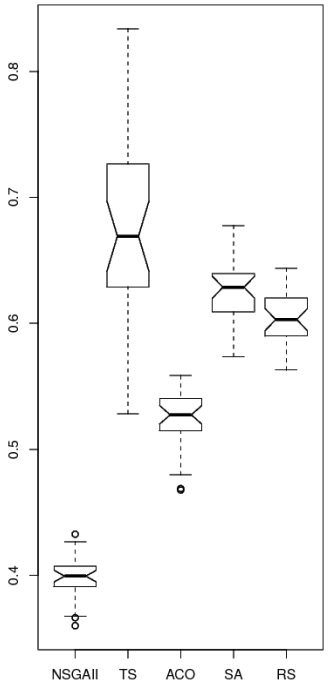
Abbildung 37: Scatterplot für nrp-g1



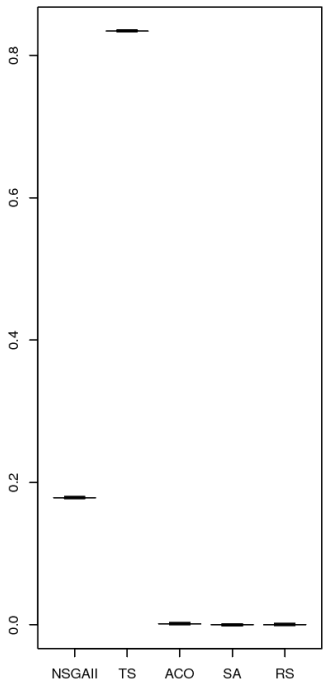
nrp-g1: Time(s)



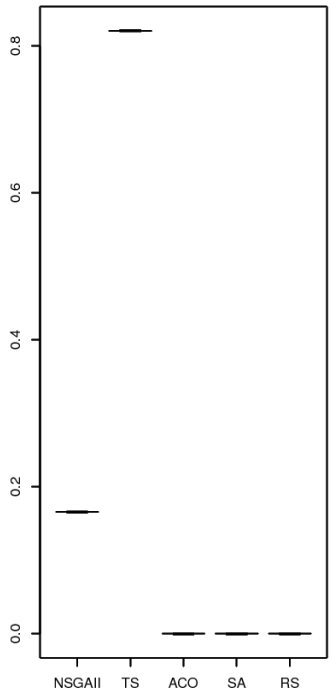
nrp-g1: Conv



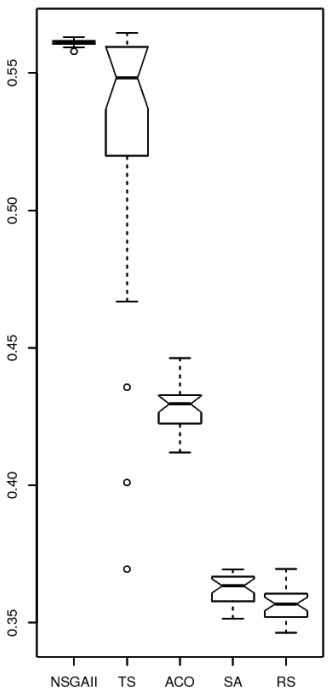
nrp-g1: Diversity



nrp-g1: Contrib



nrp-g1: UContrib



nrp-g1: HVol

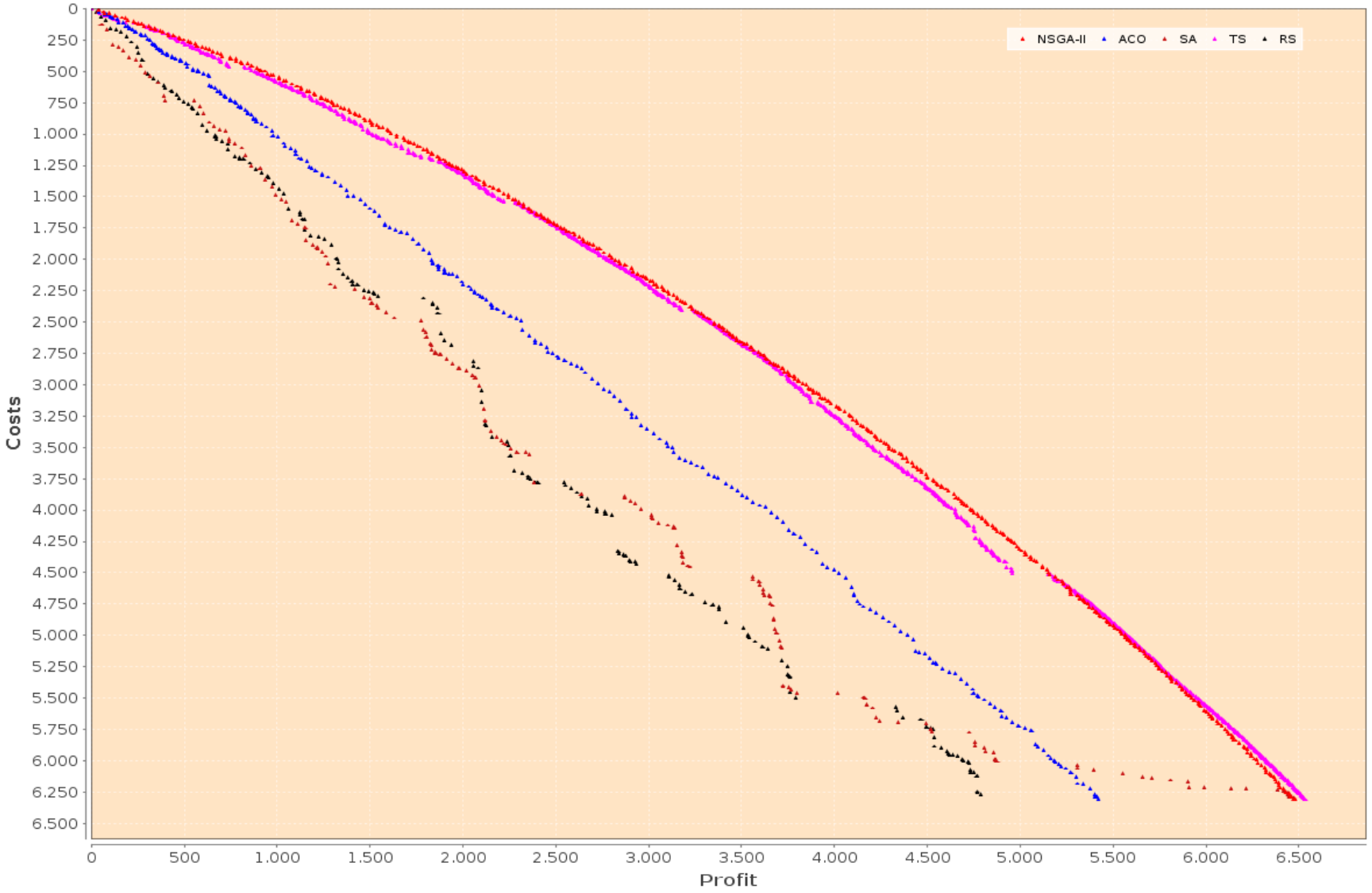
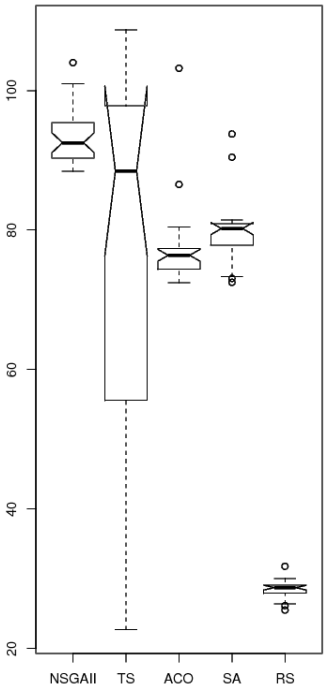
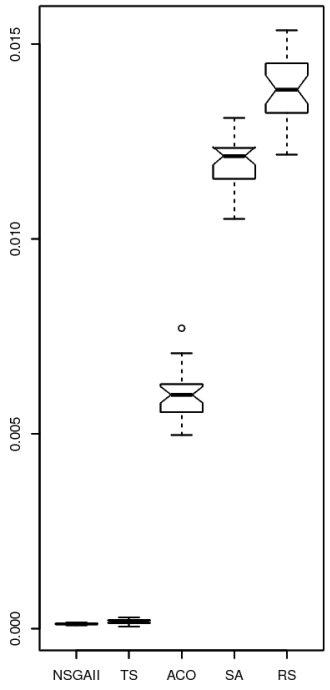


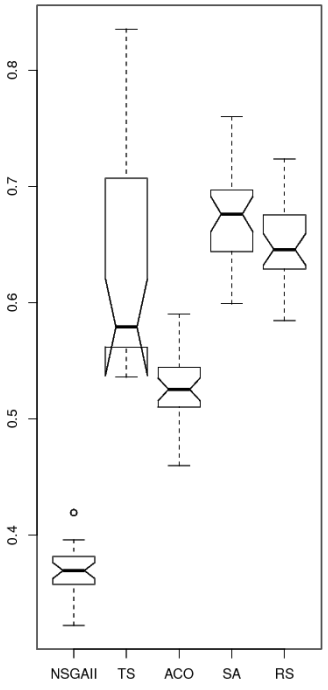
Abbildung 38: Scatterplot für nrp-g2



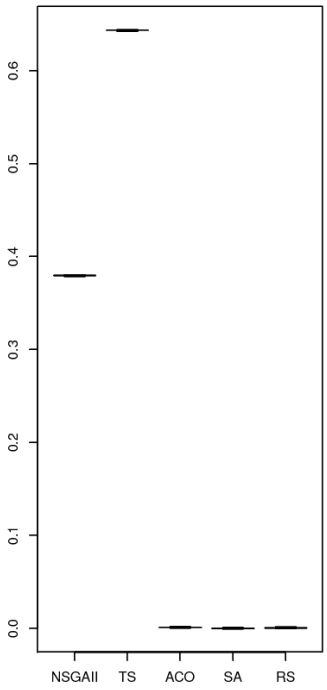
nrp-g2: Time(s)



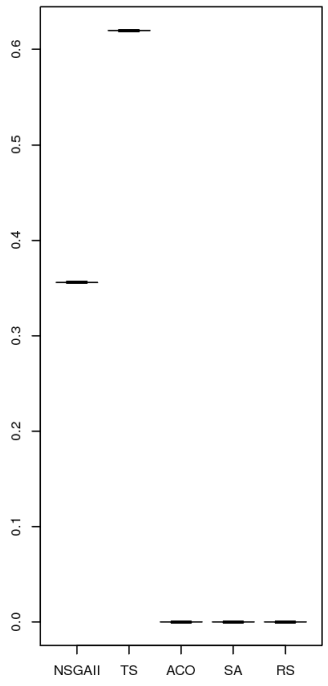
nrp-g2: Conv



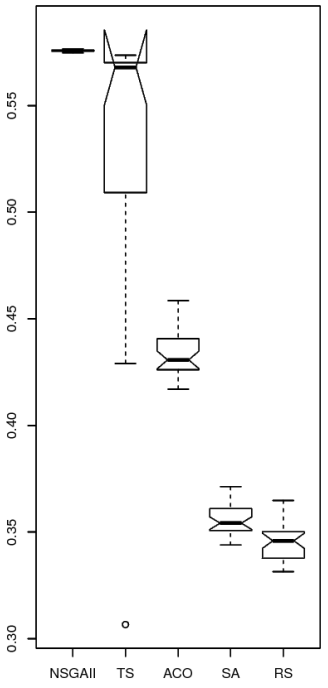
nrp-g2: Diversity



nrp-g2: Contrib



nrp-g2: UContrib



nrp-g2: HVol

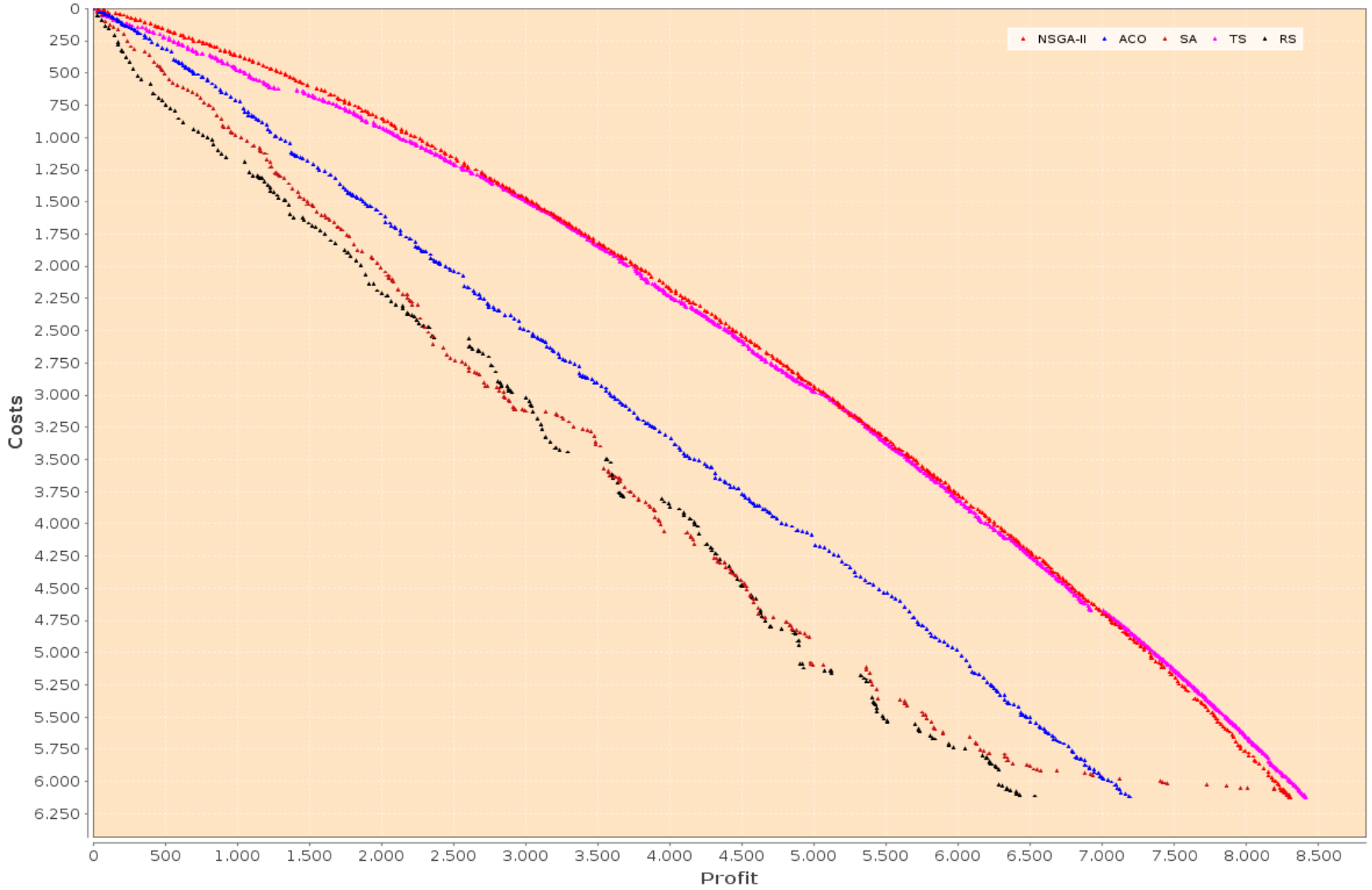
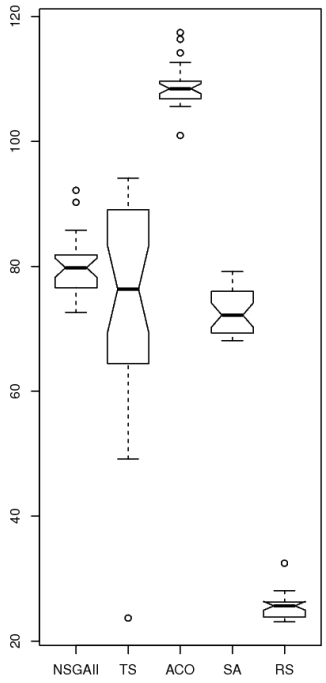
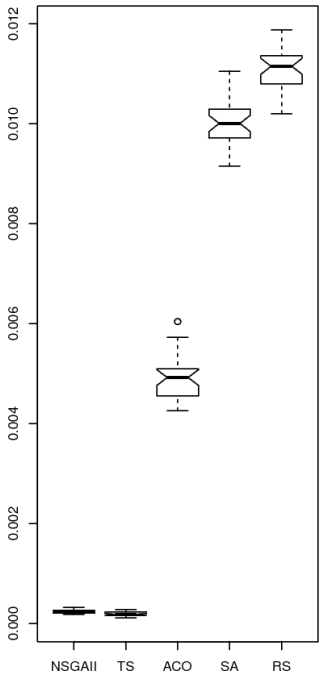


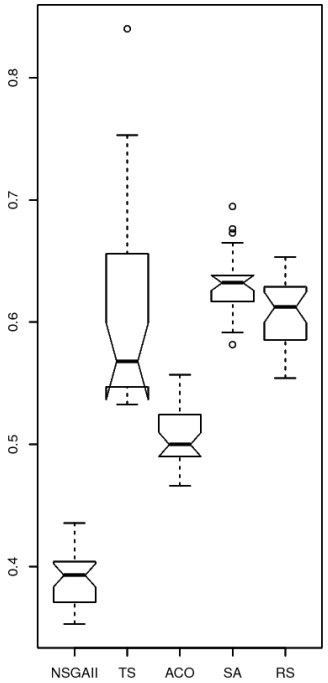
Abbildung 39: Scatterplot für nrp-g3



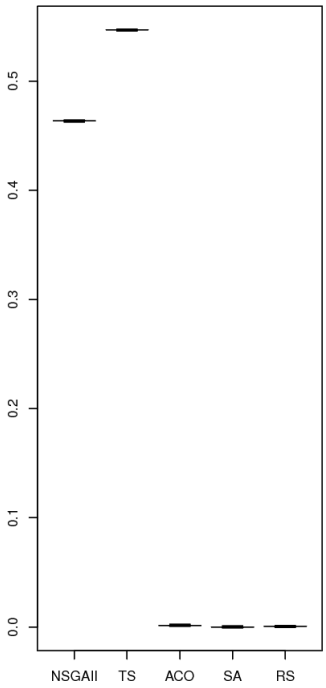
nrp-g3: Time(s)



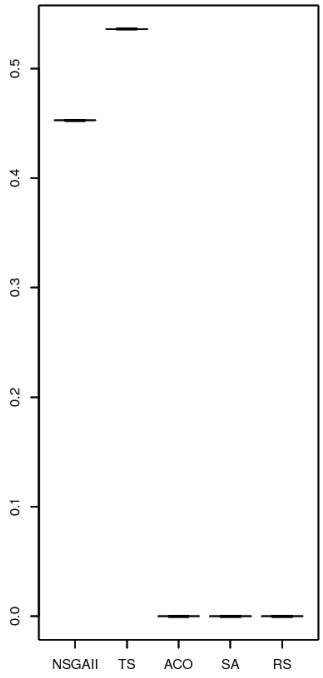
nrp-g3: Conv



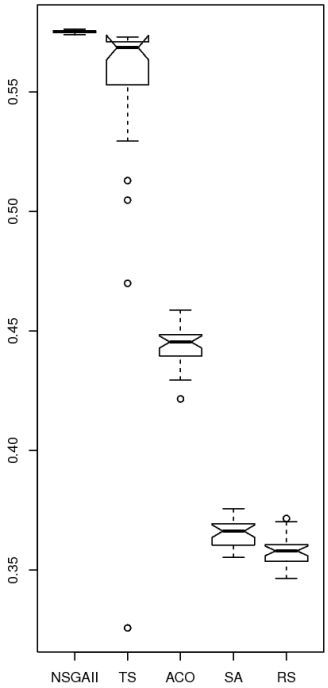
nrp-g3: Diversity



nrp-g3: Contrib



nrp-g3: UContrib



nrp-g3: HVol

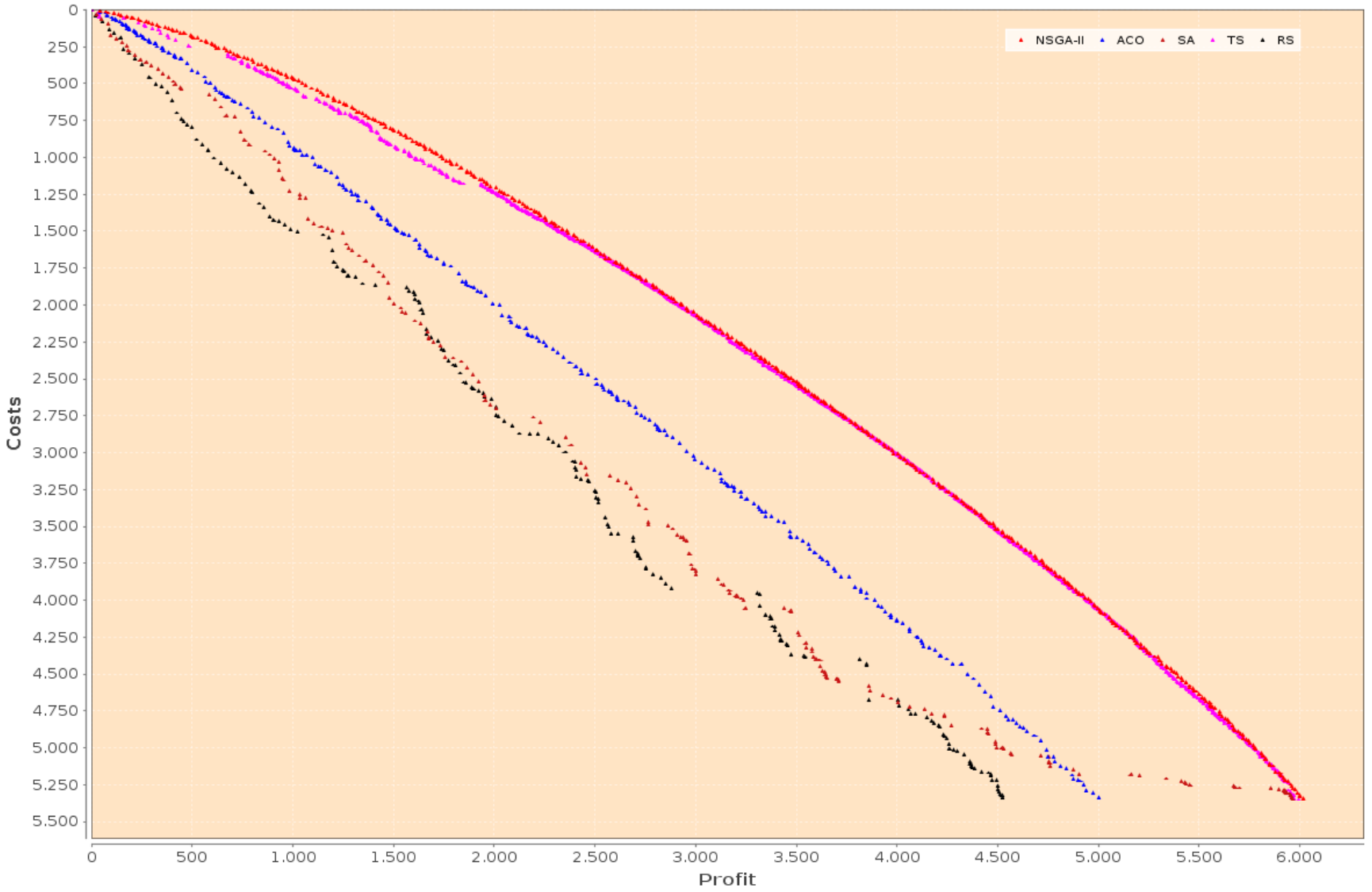
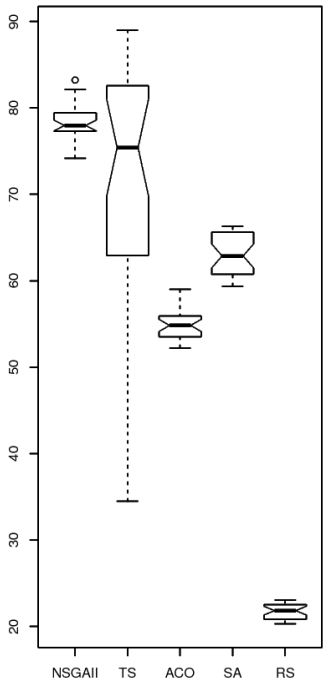
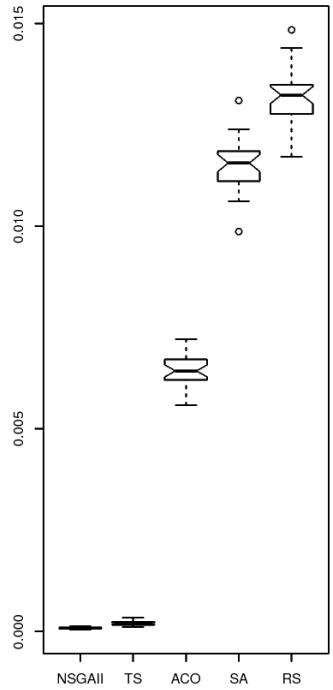


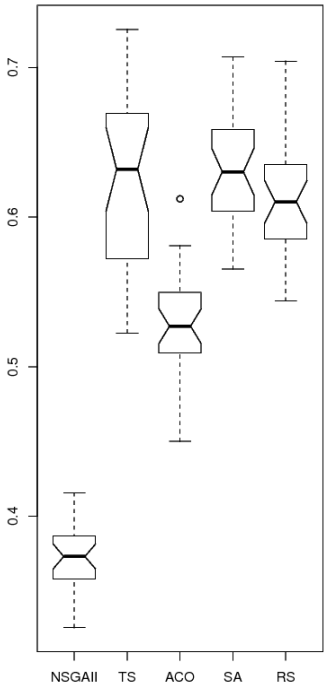
Abbildung 40: Scatterplot für nrp-g4



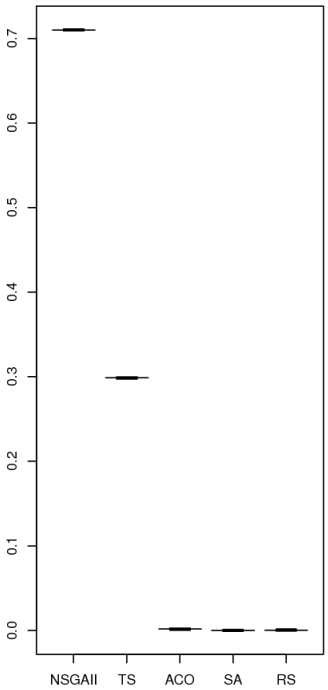
nrp-g4: Time(s)



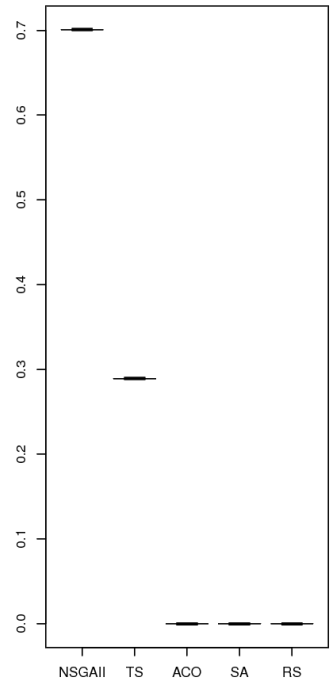
nrp-g4: Conv



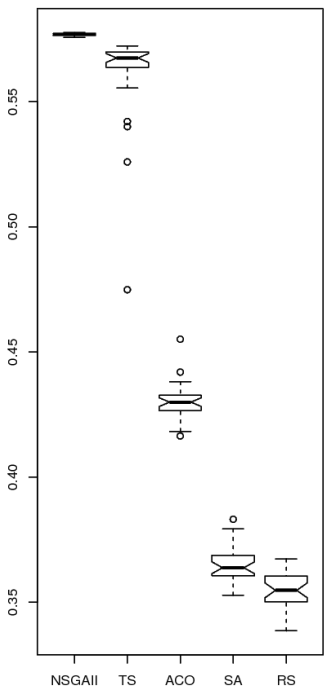
nrp-g4: Diversity



nrp-g4: Contrib



nrp-g4: UContrib



nrp-g4: HVol

7.3 Statistischer Vergleich nach Vargha-Delaney

Datensatz	Metrik		GA				ACO			TS		SA
			ACO	TS	SA	RS	TS	SA	RS	SA	RS	RS
nrp1	Qualität	Contrib	1	1	1	1	0	0	1	1	1	1
		UContrib	1	1	1	1	0.5	0.5	0.5	0.5	0.5	0.5
		Conv	1	1	1	1	0.01	1	1	1	1	1
		HVol	1	1	1	1	0.148	1	1	0.967	0.998	1
	Diversity		0	0.051	0.057	0.001	0.858	0.998	1	0.85	0.858	0.432
	Time		0	0	0	0	0.833	1	0	0.201	0.002	0
nrp2	Qualität	Contrib	1	0	1	1	0	1	1	1	1	0
		UContrib	1	0	1	1	0	0.5	0.5	1	1	0.5
		Conv	1	0.817	1	1	0	1	1	1	1	0.997
		HVol	1	0.976	1	1	0	1	1	1	1	0.977
	Diversity		0.984	1	1	1	1	1	1	0.391	0.214	0.363
	Time		1	1	1	0.696	0	0.006	0	0.96	0	0
nrp3	Qualität	Contrib	1	0	1	1	0	1	1	1	1	0
		UContib	1	0	1	1	0	0.5	0.5	1	1	0.5
		Conv	1	0.031	1	1	0	1	1	1	1	0.97
		HVol	1	0.722	1	1	0	1	1	1	1	0.946
	Diversity		1	1	1	1	1	1	1	0.847	0.838	0.393
	Time		1	0.967	0	0	0	0	0	0.033	0.033	0
nrp4	Qualität	Contrib	1	0	1	1	0	1	1	1	1	0
		UContrib	1	0	1	1	0	1	1	1	1	0.5
		Conv	1	0	1	1	0	1	1	1	1	0.934
		HVol	1	0.232	1	1	0	1	1	1	1	0.86
	Diversity		1	1	1	1	1	1	1	0.651	0.379	0.217
	Time		1	1	0.054	0	0	0	0	0	0	0
nrp5	Qualität	Contrib	1	0	1	1	0.5	0	1	1	1	0
		UContrib	1	1	1	1	0	0.5	0.5	1	1	0.5
		Conv	1	0	1	1	0	1	1	1	1	0.773
		HVol	1	0.17	1	1	0	1	1	1	1	0.502
	Diversity		0.797	1	1	1	1	1	1	0.896	0.791	0.121
	Time		1	1	0	0	0	0	0	0	0	0

Tabelle 12: Vargha-Delaney statistischer Test für klassischen Datensatz

Datensatz	Metrik		GA				ACO			TS		SA
			ACO	TS	SA	RS	TS	SA	RS	SA	RS	RS
nrp-e1	Qualität	Contrib	1	0	1	1	0	0.5	0.5	1	1	0.5
		UContrib	1	0	1	1	0	0.5	0.5	1	1	0.5
		Conv	1	0.037	1	1	0	1	1	1	1	0.878
		HVol	1	0.807	1	1	0	1	1	1	1	0.859
	Diversity		0.994	1	1	1	1	1	1	0.69	0.581	0.338
	Time		1	0.667	0.143	0	0	0	0	0.217	0	0
nrp-e2	Qualität	Contrib	1	0	1	1	0	1	1	1	1	0.5
		UContrib	1	0	1	1	0	0.5	0.5	1	1	0.5
		Conv	1	0.088	1	1	0	1	1	1	1	0.99
		HVol	1	0.881	1	1	0	1	1	1	1	0.882
	Diversity		0.954	1	1	1	1	1	1	0.5	0.308	0.233
	Time		1	0.489	0.831	0	0.032	0	0	0.723	0	0
nrp-e3	Qualität	Contrib	1	0	1	1	0	1	1	1	1	0
		UContib	1	0	1	1	0	1	1	1	1	0
		Conv	1	0.423	1	1	0	1	1	1	1	0.956
		HVol	1	0.999	1	1	0	1	1	1	1	0.809
	Diversity		0.991	1	1	1	1	1	1	0.143	0.063	0.258
	Time		1	0.374	0.386	0	0.007	0	0	0.563	0	0
nrp-e4	Qualität	Contrib	1	1	1	1	0	1	1	1	1	0
		UContrib	1	1	1	1	0	0.5	0.5	1	1	0.5
		Conv	1	0.696	1	1	0	1	1	1	1	0.993
		HVol	1	1	1	1	0.034	1	1	0.967	0.967	0.5
	Diversity		1	1	1	1	1	1	1	0.342	0.247	0.347
	Time		0.811	0.192	0.031	0	0.062	0	0	0.381	0.033	0

Tabelle 13: Vargha-Delaney statistischer Test für Eclipse-Datensatz

Datensatz	Metrik		GA				ACO			TS		SA
			ACO	TS	SA	RS	TS	SA	RS	SA	RS	RS
nrp-m1	Qualität	Contrib	1	0	1	1	0	1	1	1	1	0
		UContrib	1	0	1	1	0	1	1	1	1	0.5
		Conv	1	0	1	1	0	1	1	1	1	0.952
		HVol	1	0.99	1	1	0	1	1	1	1	0.839
	Diversity		1	1	1	1	1	1	1	0.782	0.364	0.5
	Time		1	0.168	0.039	0	0	0	0	0.454	0	0
nrp-m2	Qualität	Contrib	1	0	1	1	0	1	1	1	1	0.5
		UContrib	1	0	1	1	0	1	1	1	1	0.5
		Conv	1	0.032	1	1	0	1	1	1	1	0.96
		HVol	1	1	1	1	0	1	1	1	1	0.918
	Diversity		1	1	1	1	1	1	1	0.829	0.563	0.139
	Time		1	0.048	0.056	0	0	0	0	0.293	0	0
nrp-m3	Qualität	Contrib	1	0	1	1	0	1	0.5	1	1	0
		UContib	1	0	1	1	0	0.5	0.5	1	1	0.5
		Conv	1	0	1	1	0	1	1	1	1	0.918
		HVol	1	0.904	1	1	0	1	1	1	1	0.5
	Diversity		1	1	1	1	1	1	1	0.401	0.171	0.092
	Time		1	0.026	0.003	0	0	0	0	0.564	0	0
nrp-m4	Qualität	Contrib	1	0	1	1	0	1	1	1	1	0
		UContrib	1	0	1	1	0	0.5	0.5	1	1	0.5
		Conv	1	0.227	1	1	0	1	1	1	1	0.967
		HVol	1	1	1	1	0.033	1	1	1	1	0.773
	Diversity		1	1	1	1	1	1	1	0.783	0.617	0.148
	Time		1	0.007	0.009	0	0	0	0	0.366	0	0

Tabelle 14: Vargha-Delaney statistischer Test für Mozilla-Datensatz

Datensatz	Metrik		GA				ACO			TS		SA
			ACO	TS	SA	RS	TS	SA	RS	SA	RS	RS
nrp-g1	Qualität	Contrib	1	0	1	1	0	1	1	1	1	0
		UContrib	1	0	1	1	0	0.5	0.5	1	1	0.5
		Conv	1	0	1	1	0	1	1	1	1	0.958
		HVol	1	0.859	1	1	0.071	1	1	1	0.999	0.752
	Diversity		1	1	1	1	0.983	1	1	0.259	0.12	0.251
	Time		1	0.676	0.234	0	0.251	0	0	0.312	0	0
nrp-g2	Qualität	Contrib	1	0	1	1	0	1	1	1	1	0
		UContrib	1	0	1	1	0	0.5	0.5	1	1	0.5
		Conv	1	0.818	1	1	0	1	1	1	1	0.977
		HVol	1	1	1	1	0.067	1	1	0.967	0.967	0.831
	Diversity		1	1	1	1	0.93	1	0.999	0.738	0.697	0.333
	Time		0.032	0.363	0.031	0	0.596	0.753	0	0.427	0.034	0
nrp-g3	Qualität	Contrib	1	0	1	1	0	1	1	1	1	0
		UContib	1	0	1	1	0	0.5	0.5	1	1	0.5
		Conv	1	0.202	1	1	0	1	1	1	1	0.96
		HVol	1	1	1	1	0.033	1	1	0.967	0.967	0.822
	Diversity		1	1	1	1	0.958	1	0.999	0.713	0.65	0.29
	Time		1	0.443	0.09	0	0	0	0	0.381	0.028	0
nrp-g4	Qualität	Contrib	1	1	1	1	0	1	1	1	1	0
		UContrib	1	1	1	1	0	0.5	0.5	1	1	0.5
		Conv	1	0.994	1	1	0	1	1	1	1	0.974
		HVol	1	1	1	1	0	1	1	1	1	0.847
	Diversity		1	1	1	1	0.942	0.982	0.965	0.534	0.437	0.386
	Time		0	0.411	0	0	0.861	1	0	0.262	0	0

Tabelle 15: Vargha-Delaney statistischer Test für Gnome-Datensatz

7.4 Rangkorrelation nach Kendall

Alg.	Best-Korrelation Kendall					
	Contrib	UContrib	Conv	HVol	Diversity	Time
GA	-0.18	-0.15	0.17	0.14	-0.24	0.86
ACO	-0.02	0.42	-0.14	-0.17	-0.01	0.41
TS	0.15	0.18	0.02	0.13	0.26	0.49
SA	0.20	-	-0.21	-0.10	0.21	0.77
RS	-0.30	-	-0.26	-0.05	0.02	0.85

Tabelle 16: Kendall: bester Wert in Abhängigkeit von der Anzahl der Anforderungen

Alg.	Mittelwert-Korrelation Kendall					
	Contrib	UContrib	Conv	HVol	Diversity	Time
GA	-0.18	-0.15	0.15	0.14	-0.23	0.91
ACO	-0.02	0.42	-0.23	-0.18	-0.20	0.39
TS	0.15	0.18	-0.17	0.29	0.08	0.55
SA	0.20	-	-0.24	-0.10	0.05	0.78
RS	-0.30	-	-0.33	-0.04	-0.05	0.86

Tabelle 17: Kendall: Mittelwert in Abhängigkeit von der Anzahl der Anforderungen

Alg.	Median-Korrelation Kendall					
	Contrib	UContrib	Conv	HVol	Diversity	Time
GA	-0.18	-0.15	0.15	0.14	-0.23	0.89
ACO	-0.022	0.42	-0.21	-0.18	-0.10	0.39
TS	0.15	0.18	-0.14	0.15	0.04	0.55
SA	0.20	-	-0.26	-0.10	0.04	0.78
RS	-0.30	-	-0.32	-0.04	-0.04	0.86

Tabelle 18: Kendall: Median in Abhängigkeit von der Anzahl der Anforderungen

7.5 Rangkorrelation nach Spearman

Alg.	Best-Korrelation Spearman					
	Contrib	UContrib	Conv	HVol	Diversity	Time
GA	-0.25	-0.22	0.19	0.20	-0.36	0.97
ACO	-0.01	0.52	-0.22	-0.13	-0.02	0.53
TS	0.22	0.25	-0.01	0.15	0.31	0.66
SA	0.22	-	-0.31	0.11	0.19	0.77
RS	-0.43	-	-0.37	0.12	-0.03	0.92

Tabelle 19: Spearman: Bester Wert in Abhängigkeit von der Anzahl der Anforderungen

Alg.	Mittelwert-Korrelation Spearman					
	Contrib	UContrib	Conv	HVol	Diversity	Time
GA	-0.25	-0.22	0.17	0.18	-0.37	0.98
ACO	-0.01	0.52	-0.39	-0.13	-0.26	0.50
TS	0.22	0.25	-0.20	0.40	0.11	0.68
SA	0.22	-	-0.35	0.11	-0.05	0.77
RS	-0.43	-	-0.43	0.15	-0.17	0.94

Tabelle 20: Spearman: Mittelwert in Abhängigkeit von der Anzahl der Anforderungen

Alg.	Median-Korrelation Spearman					
	Contrib	UContrib	Conv	HVol	Diversity	Time
GA	-0.25	-0.22	0.17	0.18	-0.38	0.97
ACO	-0.01	0.52	-0.38	-0.14	-0.12	0.50
TS	0.22	0.25	-0.16	0.24	0.06	0.68
SA	0.22	-	-0.37	0.11	-0.09	0.77
RS	-0.43	-	-0.42	0.15	-0.17	0.94

Tabelle 21: Spearman: Median in Abhängigkeit von der Anzahl der Anforderungen

8 Empirische Untersuchung der Forschungsfragen

Bei kombinatorischen Algorithmen ist eine empirische Studie, welche auf Experimenten basiert, ein Quasi-Standard, um die Leistung der Algorithmen zu untersuchen. In diesem Kapitel werden fünf Forschungsfragen formuliert, die auf Grundlage der empirischen Studie beantwortet werden. Die erste und zweite Forschungsfrage befassen sich mit der Qualität von Nicht-Pareto- als auch Pareto-Algorithmen. Danach werden weitere Forschungsfragen zu Diversität, Zeit und Skalierung der Pareto-Algorithmen behandelt.

Bevor aber zu den Forschungsfragen und ihren Antworten übergegangen wird, bedarf es einer Anmerkung zur Interpretation der Tabellen für Pareto-Algorithmen. Bei Vargha-Delaney-$\hat{A}_{12}$ wurden die Werte der Metriken Time, Convergence und Diversity mit 1 - $\hat{A}_{12}$ normalisiert, sodass für alle Metriken gilt, dass ein höherer Wert der $\hat{A}_{12}$ eine bessere Leistung des Algorithmus A im Vergleich zu Algorithmus B anzeigt. Bei nichtnormalisierter Rangkorrelation nach Spearman und Kendall sind aber zwei Fälle zu unterscheiden. Die Metriken Contribution, Unique Contribution und Hypervolumen zeigen ein besseres Ergebnis der Algorithmen, je höher ihre Werte sind. Bei den Metriken Time, Convergence sowie Diversity gilt das Gegenteil. Sie zeigen ein besseres Ergebnis, je kleiner ihre Werte sind. Diese Tatsache hat Auswirkungen auf die Interpretation der Rangkorrelation nach Spearman und Kendall. Eine positive Korrelation bei Contribution, Unique Contribution und Hypervolumen bedeutet, dass sich mit steigender Anzahl der Anforderungen die Metriken besser verhalten, die negative Korrelation bedeutet dementsprechend, dass die Metriken schlechter werden. Bei Time, Convergence sowie Diversity gilt das Gegenteil. Die positive Korrelation bei diesen Metriken bedeutet, dass die Leistung mit steigender Anzahl der Anforderungen schlechter wird. Die negative Korrelation bedeutet, dass die Leistung mit steigender Anzahl der Anforderungen besser wird.

8.1 Forschungsfragen

FF1: Qualität der Nicht-Pareto-Algorithmen: Es soll untersucht werden wie die Qualität der Nicht-Pareto-Algorithmen bei NRP-Problemen ausfällt. Die Qualität wird dabei anhand des durchschnittlichen Gewinns bestimmt.

FF2: Qualität der Pareto-Algorithmen: Es soll untersucht werden, welcher Algorithmus sich am besten auf realistischen und klassischen Datensätzen gemäß den vorgestellten Qualitätskriterien verhält. Um diese Fragestellung zu beantworten, wird die Vargha-Delaney-$\hat{A}_{12}$ Methode benutzt, um jeweils zwei Algorithmen anhand der Qualitätskriterien miteinander zu vergleichen. Die Qualität steht vom Standpunkt des Entscheidungsträgers aus vor Diversität und Zeit.

FF3: Diversität der Pareto-Algorithmen: Welche Diversität der Lösungen liefert ein Algorithmus? Diese Arbeit nutzt für den Vergleich der Diversität den statistischen Vargha-Delaney-$\hat{A}_{12}$ Test. Die Messung der Diversität hat folgenden Hintergrund. Im Extremfall kann das Resultat einer Berechnung aus einer einzigen Lösung mit einer hohen Qualität bestehen, aber überhaupt keine Diversität aufweisen. Das hat zur Folge, dass dem Entscheidungsträger keine Wahl bliebe als diese Lösung zu benutzen. Deswegen erhält Diversität erst eine Bedeutung, wenn eine ausreichende Qualität der Daten gewährleistet ist.

FF4: Zeit der Pareto-Algorithmen: Wie schnell kann ein Algorithmus Lösungen produzieren? Neben Qualität und Diversität sind auch zeitliche Ressourcen, die zur Berechnung der Lösungen notwendig waren, von Bedeutung. Ein Algorithmus, der Lösungen der mittleren Qualität produziert, diese aber in sehr kurzer Zeit berechnen kann, hat einen anderen Anwendungsbereich als ein Algorithmus, welcher deutlich längere Zeit zur Berechnung von sehr guten Lösungen benötigt. Im ersten Fall kann ein schneller Algorithmus dort eingesetzt werden, wo der Entscheidungsträger temporäre Lösungen berechnen möchte, um sie in Realzeit vergleichen zu können. In diesem Szenario hat Zeit Vorrang vor Qualität, da die errechnete Qualität für eine schnelle Auswertung der Ergebnisse ausreichend ist. Der zweite Fall mit längerer Berechnungszeit bringt eine höhere Qualität der Daten, was einen größeren Nutzen in dem Bereich ergibt, wo eine wichtige Entscheidung, z. B. zur Release-Planung, getroffen werde muss. In diesem Fall liegt die Qualität der Daten vor der Laufzeit des Algorithmus.

FF5: Skalierung der Pareto-Algorithmen: Wie ist die Skalierbarkeit der Algorithmen in Bezug auf die Qualität, Diversität und Zeit? Ein Algorithmus, der Lösungen von guter Qualität und Diversität sowie in akzeptabler Zeit für keine Problemgrößen liefert, kann sich schlecht auf große Probleminstanzen skalieren. Ein weiterer wichtiger Punkt liegt darin, dass es zu einer Verschlechterung der Qualität und Diversität der Lösungen aufgrund der schlechten Skalierung des Algorithmus kommen kann, wenn die Größe des Problems gemessen an der Anzahl der Anforderungen steigt. Um diese Frage zu beantworten, werden Rangkorrelationsmethoden angewandt, um die Korrelation zwischen der Problemgröße und der Qualität, Diversität sowie Zeit zu berechnen. Für NP-harte und speziell Release-Planungs-Probleme ist davon auszugehen, dass sich der Rechenaufwand gemessen an der Zeit, sich proportional zur Problemgröße verhält [2].

8.2 Resultate zu Forschungsfragen

FF1: Qualität der Nicht-Pareto-Algorithmen: Die Tabellen 4 und 6 beschreiben Resultate der Nicht-Pareto-Experimente. In der Tabelle 4 sind Resultate der Experimente für klassische Datensätze zu sehen. Die zwei linken Spalten geben den Datensatz-Namen sowie die Kostengrenze bei dem angegebenen Kostenfaktor an. Weiter folgen fünf Algorithmen mit jeweils bestem Wert, Mittelwert und Zeit. Beispielsweise ist in der Tabelle 4 in der ersten Zeile für den Datensatz nrp-1.0.3 mit Kostenfaktor 0.3 und Kostengrenze 257 zu sehen, dass der maximale Wert des GA 1282, der Mittelwert 1265 und die Zeit 6.73 Sekunden betragen. Die Tabelle 6 beinhaltet die Resultate der Experimente für realistische Datensätze. Beispielsweise ist in der Tabelle 6 zu erkennen, dass beim Datensatz nrp-e1-0.5 mit der Kostengrenze 6557 TS den maximalen Wert 10930, den Mittelwert 10860 in 217.5 Sekunden berechnet. In der gleichen Zeile ist zu sehen, dass SA beim gleichen Datensatz den maximalen Wert 10900, den Mittelwert 10850 in 384.6 Sekunden erzielt. Ähnlich dazu lassen sich alle anderen Daten interpretieren.

Der Vergleich der absoluten Werte aus den Tabellen 4 und 6 ist aber umständlich. Aus diesem Grund hat sich diese Arbeit bei der Auswertung der Daten an der Arbeit von Xuan et al. [1] orientiert, sodass aus diesen beiden Tabellen die weiteren Tabellen 5 und 7 entstanden sind. Die Tabellen 5 und 7 stellen einen direkten prozentualen Vergleich der Leistung zweier Algorithmen mit der Formel, welche in Kapitel 6 beschrieben wurde, dar. In der Tabelle 5 bezeichnet die erste Zeile die Algorithmen GA, ACO, TS und SA, welche mit anderen Algorithmen verglichen werden. Bei fünf Algorithmen benötigt man 10 Vergleiche. Die Zeile darunter mit den Algorithmennamen und einem %-Zeichen stellt jeweils den Leistungsunterschied in Prozent zu diesem Algorithmus dar. Beispielsweise sieht man in der ersten Zeile der Tabelle 5 für den Datensatz nrp-1.0.3 den Namen: „GA“, dass GA mit vier anderen Algorithmen, ACO, TS, SA und RS, verglichen wird. In der Zeile darunter mit der Bezeichnung „ACO%“ steht der Wert 4.63. Das bedeutet, dass für den Datensatz nrp-1.0.3 GA einen Gewinn berechnet, der im Durchschnitt um 4.63 % höher als bei ACO ist. In der gleichen Zeile ist zu sehen, dass GA um 1.25 % schlechtere Lösungen als TS und um 1.09 % schlechtere Lösungen als SA liefert. In der gleichen Zeile sowie Spalte mit dem Namen „RS%“ wird deutlich, dass GA im Durchschnitt Lösungen berechnet, welche um 258.35 % die Lösungen der RS an Qualität übertreffen. Weitere Daten lassen sich analog dazu interpretieren. Die Tabelle 7 mit realistischen Datensätzen ist ähnlich zu Tabelle 5 aufgebaut. In der ersten Zeile für den Datensatz nrp-e1.0.3 kann abgelesen werden, dass GA im Schnitt 48.65 % bessere Werte als ACO liefert. GA erzeugt bessere Werte, welche um 0.75 % TS, um 0.19 % SA und um 91.75 % die Qualität der RS übertreffen.

Aus den Tabellen 5 und 7 lässt sich insbesondere erkennen, dass ACO mit einem höheren Kostenfaktor z. B. 0.7 Resultate liefert, welche sehr nahe an die besten Resultate anderer Algorithmen reichen. Ist der Kostenfaktor geringer, z. B. 0.3, so erreicht ACO deutlich schlechtere Ergebnisse im Vergleich zu anderen Algorithmen. So liefert GA in der Tabelle 5 für den Datensatz nrp-3.0.3 mit Kostenfaktor 0.3 im Schnitt Werte, welche um 60.22 % höher sind als bei ACO. Bei dem gleichen Datensatz nrp-3.0.7 mit dem

Kostenfaktor 0.7 liefert GA Ergebnisse, die nur um 7.35 % höher sind als bei ACO. Diese Tendenz lässt sich bei allen Datensätzen beobachten. Das kann mit Folgendem erklärt werden: Je kleiner der Kostenfaktor, desto kürzer werden dementsprechend die Pfade der Ameisen, was dazu führt, dass eine positive Rückkoppelung nicht eintreten kann, um bessere Lösungen zu finden. Dieser Zusammenhang zwischen dem Kostenfaktor und der Qualität der Lösungen lässt sich auch bei RS beobachten. So erreicht GA beim Datensatz nrp-3.0.3 Werte, welche um 148.35 % höher sind als beim RS. Beim gleichen Datensatz nrp-3.0.7 mit Kostenfaktor 0.7 ist die Differenz deutlich geringer und beträgt 65.63 %. Bei allen anderen Metaheuristiken lässt sich dieser Zusammenhang nicht erkennen.

Folgende Tabelle 22 wurde aus den Tabellen 5 und 7 abgeleitet und zeigt das minimale und maximale Verhältnis (min/max) der Leistung zwischen jeweils zwei Algorithmen. Beispielsweise zeigt die dritte Zeile der ersten Spalte der Tabelle 22, dass GA im Vergleich zu ACO Werte für alle 17 Datensätze liefert, welche im Bereich zwischen -0.35 % und 91.54 % liegen.

GA				ACO			TS		SA
ACO%	TS%	SA%	RS%	TS%	SA%	RS%	SA%	RS%	RS%
-0.35/91.54	-2.26/1.72	-4.02/0.92	30.46/258.35	-48.90/0	-49.55/0.12	6.11/241.52	-1.82/0.78	29.95/261.86	30.07/261.30

Tabelle 22: Qualitätsspanne beim Vergleich von jeweils zwei Nicht-Pareto-Algorithmen

Wie die Tabelle 22 zeigt, liegt der prozentuale Unterschied zwischen GA, SA und TS bei allen Datensätzen bei weniger als 4.02 %. Das bedeutet, dass diese Algorithmen bei Nicht-Pareto-Problemen Lösungen von ähnlicher Qualität liefern. Somit gehört SA zu den besten Algorithmen im Nicht-Pareto-Fall. ACO nimmt den vierten Platz ein und produziert Lösungen, welche höchsten um 0.12 % besser und bis zu 91.54 % schlechter sind als Lösungen bei den drei oben genannten Algorithmen. RS produziert bei den Nicht-Pareto-Problemen sehr schlechte Ergebnisse und liegt somit auf dem Platz 5 der Algorithmen. Allgemein kann gesagt werden, dass alle vier Metaheuristiken die Leistung der RS im Schnitt zwischen 6.11 % und 261.86 % übertreffen.

Erwähnenswert ist auch die Laufzeit der Nicht-Pareto-Algorithmen. So liegen GA, TS, SA eng beieinander und haben höchstens einen 3-fachen Unterschied an Laufzeiten. Wie anhand der Tabelle 4 zu erkennen, benötigt GA für einen der größten Datensatze nrp-4.0.7 695 Sekunden, TS 504 Sekunden und SA 430 Sekunden. ACO liefert im Vergleich zu GA, TS und SA längere Laufzeiten und braucht für den genannten Datensatz 1169 Sekunden. RS berechnet die Lösungen für diesen Datensatz in 39.14 Sekunden. Was das Verhältnis zwischen dem Implementierungsaufwand und der Qualität der Lösungen bei Nicht-Pareto-Algorithmen betrifft, so liegt SA ganz vorne, dicht gefolgt von der TS. GA liefert Resultate von ähnlicher Qualität wie SA und TS, ist aber aus Sicht der Implementierung wesentlich komplexer. ACO ist sehr komplex in der Implementierung und liefert auch nicht die besten Ergebnisse. RS ist sehr einfach zu implementieren, liefert aber Lösungen von sehr schlechter Qualität.

FF2: Qualität der Pareto-Algorithmen: Tabellen 8 – 11 präsentieren die Leistung der Pareto-Algorithmen auf klassischen sowie realistischen Datensätzen. Zu jedem Algorithmus wurde ein Mittelwert sowie Median der Qualitätskriterien, Diversität und Zeit berechnet. Die Tabelle 8 zeigt die Resultate zu klassischen Datensätzen. Für den Datensatz nrp2 berechnet NSGA-II den Mittelwert der Contribution von 4,21e-01 = 0,421, was 42,1 % beträgt. ACO liefert 0,167 %, TS liefert 58,2 %, SA liefert 0, RS liefert 0,0835 %. Es ist zu sehen, dass TS mit 58,2 % und NSGA-II mit 42,1 % den größten Beitrag liefern. Zu beachten ist, dass der gesamte Beitrag aller Algorithmen 100 % aufgrund der Tatsache übersteigen kann, dass Algorithmen gleiche Lösungen produzieren können, welche auf der Referenz-Front liegen, sodass eine Lösung mehrfach für unterschiedliche Metaheuristiken gezählt wird. Beispielsweise wird die Lösung mit dem Fitnesswert (0,0) beinah immer von allen Algorithmen gefunden. Diese Lösung wird zum Beitrag jeder einzelnen Metaheuristik gezählt. In dem angeführten Beispiel beträgt die Summe aller Beiträge 1,005505 = 100,5505 %, was eine Abweichung von 0,5505 % darstellt. Ein weiterer Grund für die Abweichung kann ein Rundungsfehler bei der Berechnung der Metrik sein. Eine Zeile tiefer ist beim gleichen Datensatz nrp2 die Unique Contribution zu sehen. Für NSGA-II beträgt der einzigartiger Beitrag 4,18e-01 = 0,418 = 41,8 %, für ACO 0, für TS 57,8 %, für SA 0 und für RS 0. Die Summe der einzigartigen Lösungen beträgt 0,996 = 99,6 %. Das bedeutet, dass 0,004 = 0,4 % der Lösungen, welche zur Referenz-Front gehören, von mehreren Algorithmen gleichzeitig gefunden und deshalb von dieser Metrik nicht erfasst wurden. Auch bei dieser Metrik bildet ein Rundungsfehler eine Möglichkeit für die Abweichung. Eine weitere Zeile tiefer beim Datensatz nrp2 ist Convergence zu sehen. Für NSGA-II beträgt die Konvergenz 9,70e-04 = 0,00097, für ACO 0,0162, für TS 0,00131, für SA 0,0301, für RS 0,0382. Da die Konvergenz den euklidischen Abstand misst, so sind Lösungen mit kleineren Werten besser. Deutlich ist, dass NSGA-II mit 0,00097 den besten Wert und TS mit 0,00131 den zweitbesten Wert liefern. Diese Tatsache wurde in den Tabellen 8 – 11 mit dunkelgrau hinterlegtem besten Wert sowie hellgrau hinterlegtem zweitbesten Wert verdeutlicht. Eine Zeile tiefer ist das Hypervolumen zu sehen. NSGA-II zeigt für diese Metrik den Wert 5,10e-01 = 0,51, ACO 0,313, TS 0,485, SA 0,158 und RS 0,135. Daraus ist zu erkennen, dass NSGA-II mit 0,51 den größten Wert und TS mit 0,485 den zweitgrößten Wert für Hypervolumen bilden. Alle anderen Algorithmen erzeugten kleinere Hypervolumen.

Die Contribution sowie Unique Contribution messen den gesamten Beitrag sowie den gesamten einzigartigen Beitrag des Algorithmus pro Experiment. Das bedeutet, dass nach 30 Iterationen für diese Metriken jeweils nur ein Wert entsteht. Aus den Tabellen 8 – 11 lässt sich ablesen, dass ACO, SA sowie RS beinah keinen Beitrag zur Erstellung der Referenz-Front leisten. Entweder ist der Beitrag gleich 0 oder diese Algorithmen finden nur wenige Lösungen. Auch in Scatterplots zu den entsprechenden Datensätzen ist zu erkennen, dass die Referenz-Front hauptsächlich von NSGA-II und TS gebildet wird. Die Abbildung 29 zeigt den Scatterplot mit den Pareto-optimalen Lösungsmengen aller Algorithmen zu dem nrp-m1-Datensatz. Dabei wird NSGA-II rot, TS rosa, ACO blau, SA braun und RS schwarz abgebildet. Die Referenz-Front wird durch Lösungen hauptsächlich von TS sowie NSGA-II gebildet. Unter dem Scatterplot sind sechs Boxplots zu den jeweiligen Metriken zu sehen. Beispielsweise zeigen alle Algorithmen eine leichte Streuung

bei der Diversität. Auch sehen die Boxplots zu Contrib sowie UContrib beinah identisch aus. Werden aber die Tabelle 9 und der Datensatz nrp-m1 für diese Metriken angeschaut, dann zeigen sich dort geringe Unterschiede, z. B. für TS zeigt Contribution 7.94-e01 = 0.794 = 79.4 % und Unique Contribution von 79.1 %. Diese geringe Differenz zwischen den beiden Metriken kommt dadurch zustande, dass sich die POLMs der Algorithmen, welche die Referenz-Front bilden, in diesem Fall NSGA-II und TS, nur an wenigen Stellen überschneiden. Die Schnittpunkte bilden Lösungen, die nicht einzigartig sind, der Rest der POLM wird als einzigartig erfasst.

Allerdings erweist sich der Vergleich anhand der absoluten Werte zwischen den Algorithmen als zu umständlich. Aus diesem Grund präsentieren die Tabelle 12 – 15 die statistischen Resultate des direkten Algorithmenvergleichs nach Vargha-Delaney-$\hat{A}_{12}$. Tabelle 13 zeigt, dass der Kopf der Tabelle zwei Reihen der Algorithmennamen enthält. Die obere Zeile verdeutlicht den Algorithmus unter dem Vergleich, die Zeile darunter den Algorithmus, mit welchem verglichen wird. Da fünf Algorithmen miteinander verglichen werden, entstehen insgesamt 10 paarweise Vergleiche pro Datensatz und Metrik.

Zum Beispiel zeigt Datensatz nrp-e1 der Tabelle 13 in der Spalte mit der Bezeichnung „GA“, dass NSGA-II mit vier anderen Algorithmen, ACO, TS, SA und RS, verglichen wird. In der Spalte „ACO“ und der Zeile „Contrib“ steht eine 1. Das hat folgende Bedeutung: NSGA-II verhält sich 100 % besser als ACO in Bezug auf die Contribution-Metrik. Alle anderen Zeilen für diesen Datensatz außer Diversity haben eine 1, was bedeutet, dass NSGA-II in diesen Metriken in 100 % der Fälle bessere Ergebnisse erzielt. Die Diversity in der gleichen Spalte der Tabelle 13 hat den Wert 0.994. In diesem Fall verhält sich NSGA-II in 99,4 % besser als ACO in Bezug auf die Diversität. Werden NSGA-II und TS in der Tabelle 13 für den gleichen Datensatz nrp-e1 betrachtet, dann ist zu sehen, dass in den Zeilen Contribution und Unique Contribution 0 steht. Das bedeutet, dass NSGA-II in 0 % der Fälle höheren Beitrag als TS liefert. Für Convergence erreicht NSGA-II im Vergleich zu TS in 0.037 = 3.7 % bessere Resultate. Beim Hypervolumen findet sich der Wert 0.807. Das bedeutet, dass in 80,7 % der Fälle NSGA-II ein höheres Hypervolumen besitzt. Bei Time steht der Wert 0.667. Das heißt, dass in 66,7 % der Fälle NSGA-II kürzere Laufzeiten als TS besitzt.

Das Problem beim Vargha-Delaney-$\hat{A}_{12}$ Vergleich von zwei Algorithmen besteht darin, dass $\hat{A}_{12}$ nicht anzeigt, in welchem Maße ein Algorithmus A besser ist als Algorithmus B. So wird z. B. in den Scatterplots zu allen Datensätzen die Referenz-Front beinah ausschließlich von NSGA-II sowie TS gebildet, sodass ACO, SA und RS kaum zu der Referenz-Front beitragen. In den Tabellen 8 – 11 wird deutlich, dass Algorithmen, ACO, SA, und RS, einen sehr geringen Beitrag liefern. In den meisten Fällen ist es nur eine oder zwei Lösungen nahe am Ursprung. So z. B. ist Tabelle 11 und nrp-g1-Datensatz zu entnehmen, dass ACO 0,139 %, SA keinen Beitrag und RS einen Beitrag von 0.0347 % leistet. Nichtsdestotrotz zeigt der Vargha-Delaney-$\hat{A}_{12}$ im direkten Vergleich zwischen SA und RS in der Tabelle 15 für nrp-g1, dass RS in 100 % der Fälle besser als SA ist. Das ist rein formal korrekt, allerdings kann nicht daraus geschlossen werden, dass RS tatsächlich ein besseres Verhalten als SA aufweist.

Um die Interpretation der Resultate noch überschaubarer zu machen, wurde Tabelle 23 aus den Vargha-Delaney-$\hat{A}_{12}$ Tabellen 12 – 15 abgeleitet. Sie fasst die Qualität der Algorithmen zusammen, indem das Verhältnis der Vergleiche bezogen auf die Gesamtzahl angegeben wird, wo ein Algorithmus bessere Qualität der Lösungen zeigte. Es wurde davon ausgegangen, dass sich ein Algorithmus besser in Bezug auf die Qualität verhält, wenn er im direkten Vargha-Delaney-$\hat{A}_{12}$ Vergleich einen Wert > 0.5 zeigt. Wenn die Qualität der Algorithmen betrachtet wird, so kann den Vargha-Delaney-$\hat{A}_{12}$ Tabellen 12 – 15 entnommen werden, dass bei 17 Datensätzen jeder Algorithmus mit 4 anderen Algorithmen in vier Qualitätsmerkmalen, Contribution, Unique Contribution, Convergence und Hypervolume, verglichen wird. Das bedeutet insgesamt $17 \cdot 4 \cdot 4 = 272$ Vergleiche.

NSGA-II	ACO	TS	SA	RS
231/272	10/272	245/272	33/272	13/272
85%	39%	90%	12%	5%

Tabelle 23: Quantitative Bewertung der Qualität

Es ist zu sehen, dass die Leistung der TS alle anderen Algorithmen übersteigt, da er in 90 % der Fälle eine bessere Qualität liefert. Auf dem zweiten Platz liegt NSGA-II mit 85 % der besseren Vergleiche. ACO erreicht in 39 % bessere Resultate. Anders als im Nicht-Pareto-Fall liefert SA das zweitschlechteste Resultat mit 12 %. Wie erwartet, wird der letzte Platz von RS mit 5 % eingenommen. Aber obwohl TS die besten Ergebnisse in Bezug auf die Qualität erzielt, wäre es eine falsche Schlussfolgerung, TS immer der NSGA-II vorzuziehen. Der Grund besteht darin, dass die Qualitätsmetrik alle vier Qualitätskriterien misst und nicht auf einzelne Fälle eingeht. So z. B. spielt der einzigartige Beitrag als ein Teil der Qualität eine wichtige Rolle. Und obwohl TS in 13 von 17 Datensätzen einen höheren einzigartigen Beitrag aufweist, liefert NSGA-II dennoch in vier von 17 Datensätzen einen höheren einzigartigen Beitrag als TS. Bei allen anderen Algorithmen gilt diese Aussage nicht, da sie beinah keinen einzigartigen Beitrag zur Referenz-Front beisteuern.

FF3: Diversität der Pareto-Algorithmen: Wie die Qualität, so lassen sich auch absolute Werte für die Diversität aus den Tabellen 8 – 11 ablesen. Die Diversität der Algorithmen soll anhand des klassischen Datensatzes gezeigt werden. Werden die Tabelle 8 und der nrp2-Datensatz betrachtet, so liefert NSGA-II in der Zeile der Diversität den Wert 5.50e-01 = 0.55, ACO 0.603, TS 0.793, SA 0.773 und RS 0.756. Es ist zu sehen, dass NSGA-II mit 0.55 den ersten und ACO mit 0.603 den zweiten Platz einnehmen. Wird stattdessen die Diversität der realistischen Datensätze herangezogen, so ist bspw. in der Tabelle 10 für den nrp-e1-Datensatz zu sehen, dass NSGA-II 4.60e-01 = 0.46, ACO 0.52, TS 0.701, SA 0.72 und RS 0.704 hinsichtlich der Diversität erreichen. Somit liegt NSGA-II wieder auf dem ersten und ACO auf dem zweiten Platz. Auch in der Abbildung 33 ist im Scatterplot zu erkennen, dass die Diversität bzw. Streuung der Lösungen bei NSGA-II und ACO viel geringer ist als bei anderen Algorithmen. Der Boxplot für den nrp-e1-Datensatz zeigt noch deutlicher, dass sich NSGA-II unterhalb der ACO befindet. Diese

beiden Algorithmen liegen im Boxplot auch klar unterhalb aller anderen Algorithmen. Im Boxplot ist gleichfalls erkennbar, dass die Streuung der Werte zur Diversität eines Algorithmus innerhalb eines Experiments vergleichsweise gering ausfällt.

Da es aber umständlich ist, Algorithmen wie oben beschrieben zu überprüfen, so wurde Vargha-Delaney-$\hat{A}_{12}$ angewandt, um jeweils zwei Algorithmen bezogen auf ihre Diversität zu vergleichen. In den Tabellen 12 – 15 ist der direkte Vergleich der Diversität zweier Algorithmen zu sehen. Bei 17 Datensätzen wird jedes Algorithmus mit vier anderen Algorithmen hinsichtlich der Diversität verglichen, d. h., insgesamt werden 68 Vergleiche erreicht. In der Tabelle 14 ist für den nrp-m1-Datensatz zu sehen, dass die Diversität der TS verglichen mit SA 0.782 erreicht. Das bedeutet, dass für diesen Datensatz TS in 78.2 % der Fälle bessere Resultate als SA erzielt.

Genauso wie im Fall der Qualität wurde folgende Tabelle 24 aus den Vargha-Delaney-$\hat{A}_{12}$ Tabellen 12 – 15 abgeleitet. Die Tabelle zeigt das Verhältnis der besseren Diversität eines Algorithmus zu der Gesamtzahl der Vergleiche.

NSGA-II	ACO	TS	SA	RS
64/68	52/68	20/68	6/68	17/68
94%	76%	31%	9%	25%

Tabelle 24: Quantitative Bewertung der Diversität

In der Tabelle 24 ist erkennbar, dass NSGA-II mit 94 % den ersten Platz bezüglich der Diversität belegt. ACO mit 76 % folgt dicht dahinter. Alle anderen Algorithmen: TS mit 31 %, SA mit 9 % und RS mit 25 % zeigen eher eine schlechte Diversität der Daten.

Wie auch vor der Durchführung der empirischen Studie erwartet wurde, verhält sich der NSGA-II-Algorithmus in Bezug auf die Diversität am besten, da er einen Operator zur Wahrung der Diversität besitzt. Der ACO-Algorithmus zeigt hingegen bereits in seiner Basisform eine hohe Diversität und übertrifft in manchen Fällen die Diversität der NSGA-II. Wie aber aus der FF2 schon deutlich wurde, liefert ACO Lösungen von schlechter Qualität, sodass diese Lösungen in beinah 100 % der Fälle von Lösungen der NSGA-II sowie TS dominiert werden. Angesichts dieser Tatsache rückt die gute Diversität der ACO in den Hintergrund. TS liefert zwar die beste Qualität der Lösungen, verfügt aber mit 31 % über eine verhältnismäßig schlechte Diversität.

FF4: Zeit der Pareto-Algorithmen: Wie die Qualität und Diversität der Daten, so lassen sich auch absolute Werte für die Laufzeit der Algorithmen aus den Tabellen 8 – 11 ablesen. Wird die Zeit in der Tabelle 8 für den klassischen nrp2-Datensatz betrachtet, so benötigt NSGA-II durchschnittlich 3.24e+01 = 32.4, ACO 223, TS 137, SA 175 und RS 32.9 Sekunden. Aus der Tabelle 10 für den realistischen nrp-e1-Datensatz kann erkannt werden, dass die Laufzeit der NSGA-II 1.16e+02 = 116, ACO 213, TS 121, SA 106 und RS 43.6 Sekunden beträgt. Während der Konzeption der Experimente wurde das Ziel verfolgt, die Parameter so einzustellen, dass alle Algorithmen ungefähr die gleiche Zeit für die Berechnung benötigen, um die Experimente möglichst unter gleichen Bedingungen durchzuführen. Allerdings ist in dem oben angeführten Beispiel zu sehen, dass die richtige

Parameterwahl nicht immer getroffen wurde, sodass für den nrp2-Datensatz NSGA-II mit 32.4 Sekunden deutlich unter anderen Metaheuristiken liegt, aber trotzdem eine gute Qualität und Diversität der Lösungen liefert.

Wie in Kapitel 6 zur Parameterwahl der Algorithmen beschrieben, ist die Laufzeit der Algorithmen sehr von den gewählten Parametern abhängig. Aus diesem Grund sollte die Zeit im Kontext dieser Arbeit eher als ein grobes Maß dienen, um zu prüfen, dass kein Algorithmus überdurchschnittlich lange für die Berechnung der Lösungen braucht. Ein Beispiel für eine überdurchschnittlich lange Laufzeit ist der Tabelle 8 zu entnehmen. Für den klassischen nrp-5-Datensatz braucht ACO im Durchschnitt 963 Sekunden, wohingegen andere Algorithmen im Bereich bis zu 111 Sekunden liegen. Die überdurchschnittlichen Laufzeiten werden ausschließlich von ACO erreicht. Sie treten vor allem dort auf, wo die Anzahl der Kunden oder die Anzahl der Anforderungen sehr hoch ist. Wie aus den Tabellen 1 und 2 abzulesen, bildet der nrp5 mit 1000 Kunden den größten Datensatz in Bezug auf die Anzahl der Kunden. Somit werden die Ameisenpfade sehr lang, sodass ACO merklich länger für die Verarbeitung der Daten benötigt. Bei Datensätzen mit einer kleineren Anzahl an Kunden ist der Unterschied zu anderen Algorithmen nicht so extrem groß.

Wie bei den Forschungsfragen zur Qualität und Diversität, soll auch hier Vargha-Delaney-$\hat{A}_{12}$ für den direkten Vergleich der Algorithmen genutzt werden. Die Tabellen 12 – 15 dienen für den Vergleich zweier Algorithmen in Bezug auf die Laufzeit. Beispielsweise ist in der Tabelle 13 für den nrp-e1-Datensatz zu sehen, dass NSGA-II im Vergleich zu ACO 1 = 100 %, zu TS 66.7 %, zu SA 14.3 % und zu RS 0 % bessere Laufzeiten liefert. Um den Vergleich zu vereinfachen und übersichtlicher zu gestalten, wurde folgende Tabelle 25 aus den Vargha-Delaney-$\hat{A}_{12}$ Tabellen abgeleitet. Die Tabelle 25 zeigt das Verhältnis der Vergleiche mit der besseren Zeit eines Algorithmus zu der Gesamtzahl der Vergleiche. Bei 17 Datensätzen wird jeder Algorithmus mit vier anderen Algorithmen in einer Zeit-Metrik verglichen, sodass insgesamt 68 Vergleiche entstehen.

NSGA-II	ACO	TS	SA	RS
23/68	9/68	28/68	38/68	67/68
34%	13%	45%	56%	98%

Tabelle 25: Quantitative Bewertung der Zeit

In der Tabelle 25 wird erkennbar, dass RS mit 98 % das beste Resultat zeigt. SA erreicht 56 %, TS 45 %, NSGA-II 34 % und ACO nur 13 %. Da aber die Zeit lediglich nur einen Faktor ausmacht, wenn ausreichend Qualität vorhanden ist, so zeigen beide Algorithmen mit der besten Qualität der Lösungen NSGA-II und TS, dass ihre Laufzeiten mit 34 % und 45 % eng beieinanderliegen. Keiner der beiden Algorithmen erzielt besonders kurze oder besonders lange Laufzeiten. Somit hat die Laufzeit für die Wahl eines besseren Algorithmus in diesem Fall keine besondere Bedeutung.

Erwähnenswert ist an dieser Stelle die Tatsache, dass TS manchmal eine sehr große Streuung der Laufzeiten zeigt. Werden bspw. die Boxplots zu den Datensätzen nrp-g2 und nrp-m2 verglichen, so ist zu erkennen, dass im Time-Boxplot zu nrp-m2 bei TS beinah keine Streuung der Laufzeiten zu sehen ist. Im Gegensatz dazu reicht in dem

Time-Boxplot zu nrp-g2 die Streuung der Laufzeiten von 20 bis 120 Sekunden. Diesem Verhalten liegt eine Eigenart zugrunde, die auch als Bug bezeichnet werden könnte und erst durch die Begutachtung der Boxplots entdeckt wurde. Die Ursache für den Bug lag darin, dass TS eine Teilmenge der Lösungen aus der Nachbarschaft der aktuellen Lösung wählte und dann überprüfte, ob diese Lösungen in der Tabu-Liste enthalten waren. Hat sich herausgestellt, dass die gewählte Teilmenge komplett als Tabu klassifiziert war, so wurde der Algorithmus abgebrochen. Der Bugfix bestand darin zu sagen, dass weitere Nachbarn so lange überprüft werden, bis die Abbruchbedingung des Algorithmus erreicht ist. Dieser Bug wurde nach der Durchführung der Experimente entdeckt und gefixt, sodass die aktuellen Boxplots zum TS nicht vorliegen. Der Bugfix garantiert nicht, dass TS immer Fortschritte macht, insbesondere wenn die Größe der Tabu-Liste, die Anzahl der Nachbarn und Mutationsrate so gewählt sind, dass alle direkten Nachbarn als Tabu klassifiziert werden. Er sorgt aber dafür, dass der Algorithmus solange nach einem Nachbarn sucht, bis die Abbruchbedingung erreicht ist. Alle anderen Algorithmen zeigen eine geringe Streuung der Laufzeiten innerhalb eines Experiments.

FF5: Skalierung der Pareto-Algorithmen: Die Tabellen 16 – 21 geben die Rangkorrelationskoeffizienten nach Spearman und Kendall wieder. Die Rangkorrelation nach Kendall in der Tabelle 16 zeigt bei NSGA-II für Contribution -0.18, für Unique Contribution -0.15, für Convergence 0.17, für Hypervolume 0.14, für Diversity -0.24 und für Time 0.86. Die Rangkorrelation nach Spearman ergibt in der Tabelle 19 für NSGA-II die Contribution -0.25, Unique Contribution -0.22, Convergence 0.19, Hypervolume 0.20, Diversity -0.36 und Time 0.97. Die negativen Werte für Contribution -0.18 und -0.25 zeigen eine leicht negative Korrelation, was bedeutet, dass der Beitrag mit steigender Anzahl der Anforderungen kleiner wird. Das Gleiche gilt für Unique Contribution mit -0.15 und -0.22. Die Werte bei Convergence 0.17 und 0.19 ergeben eine positive Korrelation, was bei der Konvergenz heißt, dass der euklidische Abstand mit steigender Anzahl der Anforderungen größer wird. Somit bedeutet eine positive Korrelation bei Convergence eine Verschlechterung der Qualität. Hypervolumen mit 0.14 und 0.20 zeigen eine leicht positive Korrelation, was bedeutet, dass das eingenommene Hypervolumen mit steigender Anzahl der Anforderungen größer wird. Die Diversity mit -0.24 und -0.36 bildet eine negative Korrelation ab, d. h., dass die Werte kleiner werden, sodass der Algorithmus mit steigender Anzahl der Anforderungen zu einer besseren Diversität der Daten führt. Die Zeit mit 0.86 und 0.97 zeigt eine beinah perfekte positive Korrelation, sodass davon auszugehen ist, dass die Laufzeit mit steigender Anzahl der Anforderungen immer zunimmt. Wie in der FF2 gezeigt wurde, liefert TS die beste Qualität der Lösungen. Aus diesem Grund ist es angebracht, auch die Skalierbarkeit der TS näher zu untersuchen. TS skaliert ziemlich gut, wie aus den Tabellen 16 – 21 deutlich wird. So liefert TS in der Tabelle 20 für Contribution, Unique Contribution sowie Hypervolume die Werte: 0.22, 0.25 sowie 0.40. Das bedeutet, dass TS eine leicht positive Korrelation für diese Metriken aufweist. Convergence hat in der gleichen Tabelle für TS den Wert -0.20. Da aber bei der Convergence-Metrik gilt, je kleiner der Wert, desto besser, zeigt die negative Korrelation in diesem Fall, dass mit steigender Anzahl an Anforderungen der euklidische Abstand kleiner und somit besser wird. Diversity ergibt für TS mit 0.11 eine leicht positive

Korrelation, was bedeutet, dass die Diversität schlechter wird. Time zeigt mit 0.68 eine stark positive Korrelation.

Wie aber aus Tabellen 8 – 11, aber auch aus den Scatterplots zu allen Datensätzen erkennbar, wird die Referenz-Front beinah zu 100 % aus den Lösungen zweier Algorithmen, NSGA-II und TS, gebildet. Damit liegt ein Nullsummenspiel bei den Metriken Contribution und Unique Contribution vor. Das bedeutet, wenn der Beitrag eines Algorithmus höher wird, muss unausweichlich der Beitrag des zweiten Algorithmus schlechter werden. Ausgeschlossen ist der Fall, bei dem die POLMs beider Algorithmen identisch sind, was aber nie vorkommt, wie aus den Scatterplots zu allen Datensätzen erkennbar. Damit lässt sich die negative Korrelation bei Contribution und Unique Contribution von NSGA-II erklären. Wie bereits erwähnt, wird die Referenz-Front hauptsächlich von NSGA-II und TS gebildet. Aus diesem Grund sollte man die Korrelationswerte bei Contribution und Unique Contribution für die Algorithmen: ACO, SA und RS nicht überbewertet werden. Der Grund liegt darin, dass die Korrelation in diesem Fall anhand ganz weniger Lösungen gebildet wird. So kann in den Tabellen 7 – 9 erkannt werden, dass z. B. SA oft überhaupt keinen Beitrag liefert. An einigen Stellen liegt der Beitrag von SA, wie z. B. bei dem Datensatz nrp-e1, bei 0.000477, was weniger als 0,05 % des Gesamtbeitrages ausmacht. Ähnlich verhält es sich mit ACO und RS. Bei der Korrelation zwischen der Unique Contribution und der Anzahl der Anforderungen wird diese Tatsache noch deutlicher. SA und RS tragen in allen Datensätzen keine einzige Lösung zu der Unique Contribution bei. Aus diesem Grund wurden in den Rangkorrelationstabellen bei diesen zwei Algorithmen keine Daten in der Spalte Unique Contribution eingetragen, da alle Werte 0 sind. Somit hat eine positive oder negative Rangkorrelation in den Fällen von ACO, SA und RS bei den Metriken Contribution sowie Unique Contribution keine Aussagekraft. Außerdem sollte darauf hingewiesen werden, dass Contribution sowie Unique Contribution den gesamten Beitrag des Algorithmus in allen 30 Iterationen messen und somit nur ein Wert für das gesamte Experiment entsteht. Folglich sind die Werte für Contribution und Unique Contribution in den Tabellen 16 – 18 sowie 19 – 21 für den besten Wert, Mittelwert sowie die Median-Korrelation gleich.

Zusammenfassend lässt sich sagen, dass die Korrelation bei allen Algorithmen individuell begutachtet werden sollte. Keiner der Algorithmen hat eine extrem positive oder extrem negative Korrelation in Bezug auf die Qualität oder Diversität gezeigt. Bei der Zeit verhält es sich anders, dort liegen die Werte oft nahe bei 1, sodass sie eine beinah perfekte positive Korrelation bilden. Es ist aber intuitiv verständlich, dass mit steigender Anzahl der Anforderungen Algorithmen zur Berechnung der Lösungen längere Zeit benötigen.

9 Fazit und Ausblick

Das übergeordnete Ziel dieser Arbeit bestand in der Untersuchung der Release-Planungs-Probleme mit unterschiedlichen Metaheuristiken. Dazu wurden fünf Algorithmen, GA, ACO, SA, TS und RS, im jMetal-Framework implementiert und in Experimenten auf klassischen sowie realistischen Datensätzen eines NRP-Problems untersucht. Die entstandenen Resultate wurden mit geeigneten Metriken zur Messung der Güte der Lösungen und anschließend mit statistischen Untersuchungsmethoden ausgewertet. Der Umfang der Arbeit wurde durch fünf Forschungsfragen zur Qualität der Nicht-Pareto- und Pareto-Algorithmen sowie zur Diversität, Zeit und Skalierung der Pareto-Algorithmen definiert.

Die empirische Studie dieser Arbeit hat gezeigt, dass im Nicht-Pareto-Fall GA, TS und SA die besten Resultate bezüglich der Qualität der Lösungen liefern. ACO hat keine überzeugenden Ergebnisse gezeigt, da die Qualität der Lösungen sowie ihre Ausführungszeit hinter den oben genannten Metaheuristiken lag. Des Weiteren wurde festgestellt, dass ACO und RS bei Nicht-Pareto-Algorithmen im Vergleich zu anderen Algorithmen bessere Lösungen liefern, wenn der Kostenfaktor höher ist. Diese Tendenz wurde bei anderen Metaheuristiken nicht beobachtet.

Bei Pareto-Problemen mit zwei Zielfunktionen wurden allerdings starke Leistungsunterschiede zwischen den einzelnen Algorithmen gemessen. Als Resultat wurde ermittelt, dass im Pareto-Fall Tabu Search sowie NSGA-II die beste Leistung gemessen anhand der Qualitätskriterien zeigten. Tabu Search liefert in 90 % der Fälle Lösungen von besserer Qualität als alle anderen Algorithmen, NSGA-II mit 85 % folgt dicht dahinter. Die übrigen Algorithmen, SA, ACO sowie RS, lieferten Lösungen, die beinah immer von Lösungen der NSGA-II und TS dominiert werden. Bei der Auswertung der Algorithmen wurde festgestellt, obwohl SA bei Nicht-Pareto-Problemen zu den besten Algorithmen wie GA und TS gehört, liefert sie bei Pareto-Problemen die zweit schlechteste Qualität, sodass sie hinter allen Algorithmen außer RS liegt.

Wenn Algorithmen eine ausreichend gute Qualität der Lösungen erreicht haben, soll auch eine gute Diversität angestrebt werden. Aus der empirischen Studie wurde deutlich, dass zwei Algorithmen, NSGA-II und ACO, die beste Diversität zeigten. NSGA-II liegt mit 94 % deutlich vor allen anderen Algorithmen, ACO mit 76 % folgt dicht dahinter. Da aber ACO keine gute Qualität der Lösungen gezeigt hat, ist ihre gute Diversität von geringer Bedeutung. NSGA-II hingegen hat sowohl eine gute Qualität als auch gute Diversität der Daten ergeben. TS mit der besten Qualität der Lösungen zeigte nur in 31 % der Fälle eine bessere Diversität als andere Algorithmen.

Wie vor der Durchführung der Experimente zu erwarten war, zeigte RS die beste Zeit und lag mit 98 % der Fälle vor allen Metaheuristiken. Da sie aber die schlechteste Qualität lieferte, so ist auch die geringe Laufzeit in diesem Kontext irrelevant. NSGA-II und TS, welche die beste Qualität der Lösungen lieferten, erreichten mit 34 % und 45 % nur die mittleren Laufzeiten unter allen Algorithmen. ACO mit 13 % hat die längste Laufzeit und auch keine gute Qualität der Lösungen gezeigt.

Diese Arbeit hat festgestellt, dass TS gut skaliert. NSGA-II hat allerdings keine gute Skalierung des Beitrags sowie des einzigartigen Beitrags präsentiert. Ansonsten wurde bei keinem Algorithmus eine perfekte positive oder perfekte negative Korrelation bei der

Qualität oder Diversität festgestellt. Die Zeit allerdings zeigte bei allen Algorithmen eine Tendenz zur perfekten positiven Korrelation.

Hinsichtlich des Implementierungsaufwandes und der Leistung in den Pareto-Algorithmen liegt Tabu Search weit vorne. TS hat immer eine gute bis sehr gute Qualität der Ergebnisse erzielt und ist im Vergleich zu NSGA-II sehr leicht zu implementieren. Werden stattdessen ausschließlich Nicht-Pareto-Probleme betrachtet, wo nur eine Zielfunktion optimiert wird, so ist stattdessen SA zu empfehlen. Bei Nicht-Pareto-Problemen liegt die Qualität der SA auf dem Niveau von TS und GA und der Implementierungsaufwand ist noch geringer als bei TS. ACO ist aus Sicht dieser Arbeit nicht zu empfehlen, da der Implementierungsaufwand sehr hoch ist sowie die Einstellung der Parameter den höchsten Einarbeitungsaufwand unter allen Algorithmen benötigt. Die Leistung sowie die Ausführungszeit der ACO lassen auch zu wünschen übrig. Random Search lieferte immer die schlechtesten Qualität der Lösungen. Da aber RS in dieser Arbeit von Anfang an verwendet wurde, um die unterste Schranke der Leistungsfähigkeit der Algorithmen zu bilden, so kann gesagt werden, dass alle Algorithmen dieses Ziel übertroffen haben.

Nichtsdestotrotz kann durch diese empirische Studie keine allgemeingültige Aussage zur Leistungsfähigkeit der untersuchten Algorithmen getroffen werden, da zum Einen eine begrenzte Anzahl an Datensätzen ausgewertet wurde, zum Anderen spielen die Implementierung, genutzten Operatoren, Parameterwahl sowie andere Faktoren eine wichtige Rolle in der Leistungsfähigkeit der Algorithmen, sodass sie einen Bereich für weitere Untersuchungen bilden. Zukünftig kann darüber nachgedacht werden, die Verbesserungsvorschläge, welche in den einzelnen Kapiteln beschrieben wurden, bei der Implementierung der Algorithmen einzubeziehen.

10 Anhang

10.1 Evaluierung mit R

Zur statistischen Auswertung wurde R benutzt. Hier wird gezeigt wie die Daten der Experimente mit R eingelesen und mittels statistischer Tests von Vargha-Delaney-$\hat{A}_{12}$, Kendall und Spearman evaluiert werden können.

```
AMeasure <- function(a,b){

    r = rank(c(a,b))
    R1 = sum(r[seq_along(a)])

    m = length(a)
    n = length(b)
    A = (R1/m - (m+1)/2)/n

    A
}
> dataTmp = read.csv("testData.dat", sep=",", header=F)
> AMeasure(dataTmp[,2], dataTmp[,3])
[1] 0.5
```

```
> dataTmp = read.csv("testData.dat", sep=",", header=F)
> cor.test(dataTmp[,2], dataTmp[,3], method="kendall")

        Kendall's rank correlation tau

data:  dataTmp[, 2] and dataTmp[, 3]
T = 15, p-value = 0.2389
alternative hypothesis: true tau is not equal to 0
sample estimates:
      tau
0.4285714
```

```
> dataTmp = read.csv("testData.dat", sep=",", header=F)
> cor.test(dataTmp[,2], dataTmp[,3], method="spearman")

        Spearman's rank correlation rho

data:  dataTmp[, 2] and dataTmp[, 3]
S = 24, p-value = 0.2
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.5714286
```

Referenzen

[1] Xuan, Jifeng, He Jiang, Zhilei Ren, und Zhongxuan Luo. „Solving the Large Scale Next Release Problem with a Backbone-Based Multilevel Algorithm“. IEEE Transactions on Software Engineering 38, Nr. 5 (September 2012): 1195–1212. https://doi.org/10.1109/TSE.2011.92.

[2] Zhang, Yuanyuan, Mark Harman, Gabriela Ochoa, Guenther Ruhe, und Sjaak Brinkkemper. „An empirical study of meta-and hyper-heuristic search for multi-objective release planning“. RN 14, Nr. 07 (2014).

[3] Durillo, Juan J., YuanYuan Zhang, Enrique Alba, und Antonio J. Nebro. „A Study of the Multi-objective Next Release Problem“, 49–58. IEEE, 2009. https://doi.org/10.1109/SSBSE.2009.21.

[4] Branke, J., Deb, K. "Multiobjective Optimization: Interactive and Evolutionary Approaches."

[5] Bagnall, A.J., Rayward-Smith V.J., Whittley, I.M.: The Next Release Problem. Information and Software Technology 43(14), 883-890 (2001)

[6] Harman, Mark, und Phil McMinn. „Search Based Software Engineering: Techniques, Taxonomy, Tutorial“, o. J., 54.

[7] Neumann, Frank, und Carsten Witt. Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity. Natural Computing Series. Berlin: Springer, 2010.

[8] Michael R. Garey, David S. Johnson: „A Guide to the Theory of NP-Completeness“.

[9] Collette, Yann. Multiobjective Optimization: Principles and Case Studies. Berlin; London: Springer, 2011.

[10] Wassim Jaziri: „Local Search Techniques: Focus on Tabu Search“.

[11] J. Dreo, A. Petrowski, P. Siarry, E. Taillard: Metaheuristics for Hard Optimization: Springer, Berlin, ISBN 10 354023022X, 2006, 369 Pages. Mathematical Methods of Operations Research 66, Nr. 3 (21. November 2007): 557–58. https://doi.org/10.1007/s00186-007-0180-y.

[12] Zhang, Yuanyuan, Mark Harman, und S. Afshin Mansouri. „The Multi-Objective next Release Problem“, 1129. ACM Press, 2007. https://doi.org/10.1145/1276958.1277179.

[13] Rani, Kumari Seema. „Next Release Problem: Emerging trend of Requirement Engineering“ 02, Nr. 02 (2017): 8.

[14] Bagnall, A.J., V.J. Rayward-Smith, und I.M. Whittley. „The next Release Problem“. Information and Software Technology 43, Nr. 14 (Dezember 2001): 883–90. https://doi.org/10.1016/S0950-5849(01)00194-X.

[15] Durillo, Juan J., YuanYuan Zhang, Enrique Alba, und Antonio J. Nebro. „A Study of the Multi-Objective Next Release Problem“, 49–58. IEEE, 2009. https://doi.org/10.1109/SSBSE.2009.21.

[16] Sagrado, José del, Isabel María del Águila, Francisco Javier Orellana, und Samuel Túnez. „Requirements Selection: Knowledge based optimization techniques for solving the Next Release Problem.“ In KESE, 2010.

[17] Sivanandam, S. N., und S. N. Deepa. Introduction to Genetic Algorithms. Berlin; New York: Springer, 2007.

[18] Pham, D. T. Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing, Springer, 2012.

[19] Deb, K., A. Pratap, S. Agarwal, und T. Meyarivan. „A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II“. IEEE Transactions on Evolutionary Computation 6, Nr. 2 (April 2002): 182–97. https://doi.org/10.1109/4235.996017.

[20] Ah, R T F, K Deb, und H C S Rughooputh. „Comparison of NSGA-II and SPEA2 on the Multiobjective Environmental/Economic Dispatch Problem“, o. J., 27.

[21] E. Zitzler, L. Thiele: „Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach“

[22] de Souza J.T., Maia C.L.B., Ferreira T..N., Carmo R.A.F.., Brasil M.M.A. (2011) An Ant Colony Optimization Approach to the Software Release Planning with Dependent Requirements. In: Cohen M.B., Ó Cinnéide M. (eds) Search Based Software Engineering. SSBSE 2011. Lecture Notes in Computer Science, vol 6956. Springer, Berlin, Heidelberg

[23] Dorigo, Marco, Vittorio Maniezzo, und Alberto Colorni. „Ant system: optimization by a colony of cooperating agents“. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26, Nr. 1 (1996): 29–41.

[24] Dorigo, Marco, Di Caro, Gianni: „The Ant Colony Optimization Meta-Heuristic“

[25] Neumann, Frank, und Carsten Witt. Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity. Natural Computing Series. Berlin: Springer, 2010.

[26] Fabregat, Ramon, und Yezid Donoso. Multi-Objective Optimization in Computer Networks Using Metaheuristics. Auerbach Publications, 2007. https://doi.org/10.1201/9781420013627.

[27] Van Laarhoven, Peter JM, and Emile HL Aarts. Simulated annealing. Springer Netherlands, 1987.

[28] Glover, Fred, and Manuel Laguna. Tabu search. Springer US, 1999.

[29] Kanyapat Watcharasitthiwat, Paramote Wardkein, und Saravuth Pothiya. Multiple Tabu Search Algorithm for Solving the Topology Network Design. INTECH Open Access Publisher, 2008.

[30] Antonio J. Nebro and Juan J. Durillo. jMetal 5 Documentation. University of Málaga, 2015.

[31] Siwei Jiang, Yew-Soon Ong, Jie Zhang, Liang Feng: „Consistencies and Contradictions of Performance Metrics in Multiobjective Optimization“.

[32] David A. Van Veldhuizen, Gary B. Lamont: „On Measuring Multiobjective Evolutionary Algorithm Performance“.

[33] Li, Ke, und Kalyanmoy Deb. „An Evolutionary Many-Objective Optimization Algorithm Based on Dominance and Decomposition“. IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, o. J., 23.

[34] Andrea Arcuri, Lionel Briand: „A Hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering“

[35] Hellbrück, Reiner P. Angewandte Statistik mit R: eine Einführung für Ökonomen und Sozialwissenschaftler. 1. Aufl. Lehrbuch. Wiesbaden: Gabler, 2009.

[36] Durillo, Juan J, Antonio J Nebro, Francisco Luna, Bernabe Dorronsoro, und Enrique Alba. „JMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics∗“, o. J., 12.

[37] Bezerra, Leonardo C. T., Manuel López-Ibáñez, und Thomas Stützle. „An Empirical Assessment of the Properties of Inverted Generational Distance on Multi- and Many-Objective Optimization“. In Evolutionary Multi-Criterion Optimization, herausgegeben von Heike Trautmann, Günter Rudolph, Kathrin Klamroth, Oliver Schütze, Margaret Wiecek, Yaochu Jin, und Christian Grimme, 10173:31–45. Cham: Springer International Publishing, 2017.

[38] Pham, D. T. Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing. Springer, 2012.

[39] Zitzler, Eckart, und Lothar Thiele. „Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach“. IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION 3, Nr. 4 (1999): 15.

[40] Tang, K. S., Hrsg. Multiobjective Optimization Methodology: A Jumping Gene Approach. Industrial Electronics Series. Boca Raton: CRC Press, 2012.

[41] Sachs, Lothar. Statistische Methoden: mit 5 Abbildungen, 25 Tabellen und einer Klapptafel. 5., neubearb. Aufl. Berlin: Springer, 1982.

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Berlin, den 17. Juni 2018 .....