technische universität
dortmund

Parameteroptimierung des mobilen Roboters Adept Lynx

Dino Alexander Menges

Algorithm Engineering Report
TR15-2-003
Dezember 2015
ISSN 1864-4503

Fakultät für Informatik
Algorithm Engineering (LS 11)
44221 Dortmund / Germany
http://ls11-www.cs.uni-dortmund.de/

technische universität
dortmund

Diplomarbeit

Parameteroptimierung des mobilen Roboters Adept Lynx

Dino Alexander Menges

16. Juni 2015

Betreuer:
Prof. Dr. Günter Rudolph
Dipl.-Inf. Simon Wessing

Fakultät für Informatik
Algorithm Engineering (Ls11)
Technische Universität Dortmund
http://ls11-www.cs.tu-dortmund.de

Inhaltsverzeichnis

Kapitel 1

Einleitung

Diese Arbeit befasst sich mit der Parameteroptimierung des mobilen Roboters *Adept Lynx*. Im Speziellen werden die Parameter zur Pfadplanung betrachtet. Der *Adept Lynx* ist ein *Autonomous Indoor Vehicle (AIV)* - eine Untergruppe der *Fahrerlosen Transportsysteme (FTS)*. Im Gegensatz zu herkömmlichen FTS navigiert der *Adept Lynx* nicht spurgebunden. Vielmehr navigiert er autonom in einem zuvor eingelernten Gebiet.

Zur Navigation und Kollisionsvermeidung besitzt er einen in Fahrtrichtung ausgerichteten Sicherheitslaser (SICK S300), der aus einem Schlitz in einer Höhe von 200mm herausschaut (vgl. Abb. 1.1) [1]. Der Laser kann dank seines großen Öffnungswinkels von 270°, von denen allerdings nur 250° genutzt werden können, auch Hindernisse erkennen, die sich schräg hinter dem Roboter befinden. Zur Hindernisvermeidung verwendet der *Adept Lynx* den Dynamic Window Approach [16].

Abbildung 1.1: Bild eines Adept Lynx

Der Vorteil gegenüber spurgebundenen Fahrzeugen ist, dass keine Veränderungen am Einsatzgebiet vorgenommen werden müssen, z. B. Magnetstreifen am Boden, Reflektoren, etc. Der Arbeitsraum wird durch Abfahren mit Hilfe eines Joysticks kartiert, um die Orientierung zu gewährleisten. Ein Algorithmus auf Basis des *Simultaneous Localization And Mapping (SLAM)* Konzepts [24] erstellt die Karte. Diese Karte ähnelt dem Grundriss des Arbeitsbereichs (vgl. Abb. 1.2a). Der Arbeitsbereich (vgl. Abb. 1.2b) kann durch eine Vielzahl verschiedener Modifikationen beeinflusst werden; zum Beispiel können Zielpunkte, verbotene Bereiche, bevorzugte Wege, Einbahnstraßen oder Bereiche mit Links-/Rechtsverkehr definiert werden.

1.1 Motivation und Hintergrund

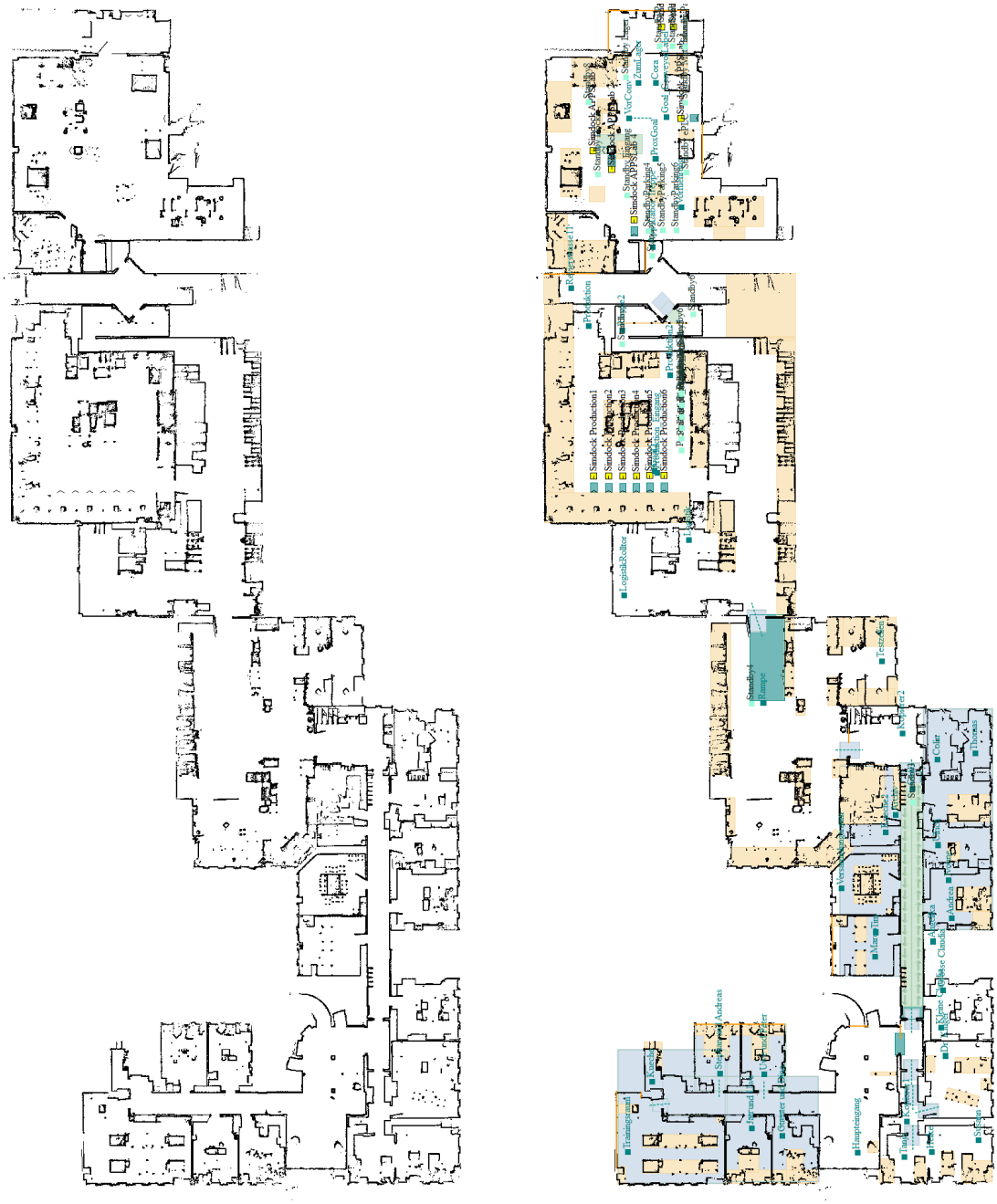
Nicht nur durch die zusätzlichen virtuellen Einflüsse ist die Lösung des Problems der Pfadplanung nicht trivial. Bei autonom navigierenden mobilen Robotern, wie dem *Adept Lynx*, beeinflussen viele Parameter den zur Pfadplanung verwendeten Algorithmus. Sie sind stark abhängig von der Umgebung, in welcher der Roboter arbeitet, sodass für verschiedene Umgebungen unterschiedliche Parametersätze eine optimale Lösung/einen optimalen Pfad liefern. Ein Pfad ist dabei ein zweidimensionaler Weg, welcher von der aktuellen Roboterposition zum angegebenen Zielpunkt führt.

1.1.1 Problembeschreibung

Die Aufgabe des Roboters ist mit der Planung des Pfades nicht beendet; vielmehr muss er dem geplanten Pfad bis zum Zielpunkt folgen. Das zu lösende Problem ist somit zweigeteilt. Es besteht aus dem Teil der Pfadplanung gefolgt von der Pfadverfolgung.

Pfadplanung

Während der Pfadplanung berechnet der Roboter den Pfad von seiner aktuellen Position zum Zielpunkt. Dieser Pfad wird als *globaler Pfad* bezeichnet. Er wird auf Basis der Karte berechnet. Das bedeutet, dass nur *statische* Hindernisse berücksichtigt werden. Statische Hindernisse sind in diesem Fall sowohl zuvor kartierte Hindernisse, als auch virtuelle Hindernisse, wie verbotene Zonen und Linien. Die Pfadplanung erfolgt auf dem Prinzip der *Kostenreduktion*. Dabei wird die Karte in ein Gitter unterteilt. Eine Gitterzelle zu durchqueren kostet einen bestimmten Preis. Der Pfad mit dem günstigsten Gesamtpreis, bestimmt durch die Summe der Zellenkosten, wird ausgegeben. Eine Zelle, die von einem Hindernis belegt oder Teil einer verbotenen Zone ist, verursacht unendlich hohe Kosten. Das Gleiche gilt, wenn Zellen einer Einbahnstraße in verkehrter Richtung durchquert werden müssten sowie auf der verkehrten Spur bei Rechts-/Linksverkehr. Freie Zellen haben konstante Standardkosten. Der Gegensatz zur Bestrafung (erhöhte Kosten) ist die bevorzugte



(a) Grundrisskarte aufgenommen vom *Adept Lynx*

(b) Grundrisskarte ausgefüllt mit Verkehrsregeln und Zielpunkten; orangene Flächen kennzeichnen verbotene Bereiche. Die grüne Fläche mit Pfeilen ist ein Bereich mit Rechtsverkehr. Grüngestrichelte Linien sind bevorzugte Wege während dunkelgrüne Quadrate Zielpunkte darstellen.

Abbildung 1.2: Karte des Adept Lynx; Rohdaten (1.2a) und ausgefüllt mit Verkehrsregeln (1.2b)

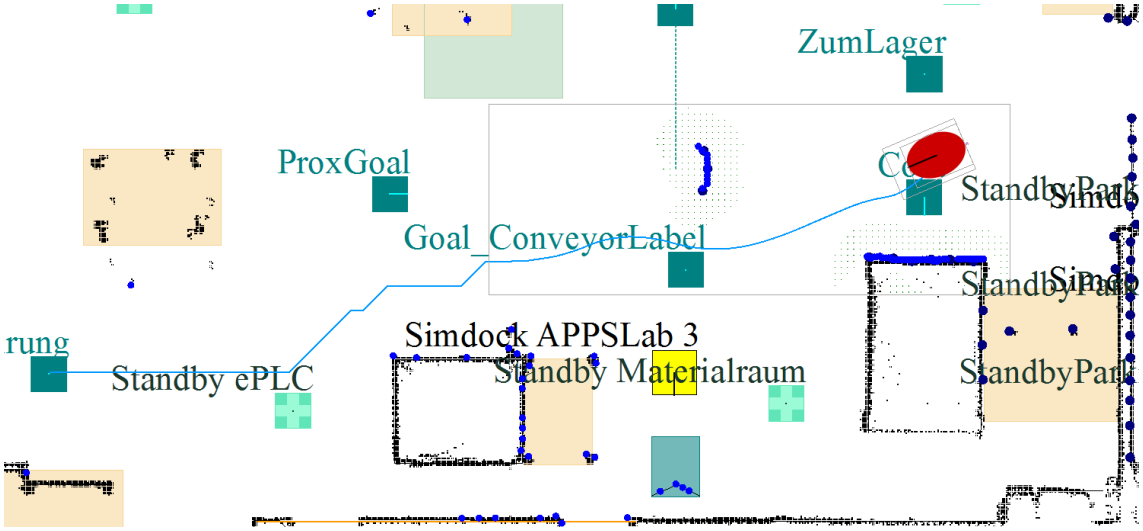


Abbildung 1.3: Globaler Pfad des *Adept Lynx* mit lokaler Korrektur zur Hindernisvermeidung

Linie. Zellen entlang bzw. auf der Linie zu durchqueren ist günstiger als die Standardkosten. Der Divisor, durch den die Standardkosten geteilt werden, kann pro Linie gesetzt werden. Es gilt: Je höher der Divisor, desto attraktiver wird es für den Roboter entlang dieser Linie zu fahren bzw. zu planen. Das bedeutet auch, dass der Roboter ggf. kleine Umwege in Kauf nimmt, damit er anstelle des direkten Weges über eine bevorzugte Linie fahren kann. Eine Pfadplanung schlägt fehl, wenn der Roboter sein Ziel nicht erreichen kann. Das heißt, dass es keinen Pfad ohne unendliche Kosten von der Roboterposition zum Zielpunkt gibt.

Pfadverfolgung

Im zweiten Teil, der Pfadverfolgung, fährt der Roboter den geplanten Pfad ab. Während der Roboter dem Pfad folgt, weicht er Hindernissen aus. Dazu gehören sowohl *statische* Hindernisse (wie bei der Pfadplanung) als auch *dynamische* Hindernisse. Dynamische Hindernisse sind hierbei Hindernisse, die vom Laserscanner erkannt werden. Diese Hindernisse müssen nicht zwangsläufig beweglich sein. Es kann sich dabei sowohl um nicht kartierte Gegenstände als auch um Fußgänger, Gabelstapler oder ähnliches handeln. Der daraus resultierende Pfad wird als *lokaler Pfad* bezeichnet. Die Bezeichnung soll deutlich machen, dass er nur einen Teil des *globalen Pfades* verändert. Das bedeutet, dass der Roboter während der Pfadverfolgung einen bestimmten Bereich des *globalen Pfades* verändern kann. Das Ende des *lokalen Pfades* liegt wieder auf dem *globalen Pfad* (vgl. Abb. 1.3). Der *lokale Pfad* kann somit als Korrektur des *globalen Pfades* zur dynamischen Hindernisvermeidung gesehen werden. Die Abbildung 1.3 stellt den *globalen Pfad* des *Adept Lynx* mit Änderungen im *lokalen Pfad* dar. Die blauen Punktewolken im Bild zeigen die aktuell gemessenen Reflektionen des Laserscanners. Diesem Hindernis weicht der *lokale Pfad* aus. Zum Ende

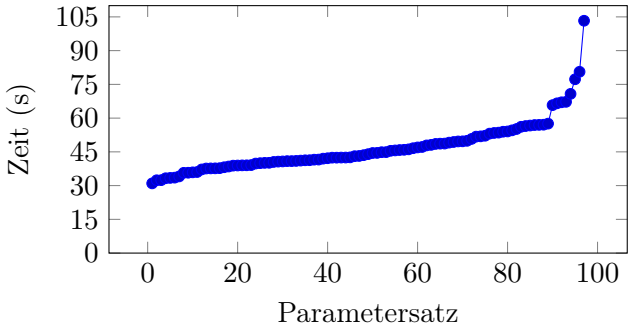


Abbildung 1.4: Dauer der Auswertung eines Individuums

mündet er in den *globalen Pfad*. Der *lokale Pfad* ist der Teil, der sich in dem großen grauen Kästchen um den Roboter befindet.

Die Aufgabe dieser Arbeit ist es, die Parameter zur Beeinflussung der Pfadplanung und Pfadverfolgung zu optimieren. Da über die Struktur des Problems nicht viel bekannt ist, übernimmt ein *Evolutionärer Algorithmus (EA)* die Optimierung. EAs haben den Vorteil, dass sie ohne Expertenwissen Probleme lösen können. Dabei generiert der EA Parametersätze, deren Güte über Zielfunktionen berechnet werden. Die Bewertung dieser Parametersätze findet auf einem simulierten Roboter statt. Der Simulator stellt damit die *Black-Box* zur Berechnung $x \rightarrow f(x)$ dar. Das hat den Vorteil, dass eine mögliche Kollision auf Grund eines schlechten Parametersatzes den Roboter nicht beschädigt. Nachdem die Optimierung abgeschlossen ist, werden die ausgegebenen Parametersätze auf einem echten Roboter verifiziert.

Da verschiedene, konfliktäre Zielfunktionen die Parameter bewerten, wird zur Optimierung ein *Evolutionärer Mehrkriterieller Optimier-Algorithmus (EMOA)* verwendet. Ausgewählt wurde der *S-Metrik-Selektion-EMOA (SMS-EMOA)* von Emmerich et al. [14]. Er hat gegenüber anderen EMOAs, wie dem *Nondominated Sorting Genetic Algorithm II (NSGAII)* [11] oder dem *Strength Pareto Evolutionary Algorithm 2 (SPEA2)* [29] den Vorteil auch bei hoch-dimensionalen Zielfunktionen eine Lösung zu finden [3]. Außerdem benötigt er, außer den für EAs üblichen, nur wenige weitere Strategieparameter. Der erhöhte Rechenaufwand, den der SMS-EMOA im Vergleich zu anderen EMOAs zur Berechnung der S-Metrik [30] benötigt, kann vernachlässigt werden. Die Berechnung der Zielfunktion mittels Simulation ist deutlich zeitintensiver. Das hängt damit zusammen, dass die Simulation in Realzeit abläuft: $t_{sim} = t_{real}$. Abbildung 1.4 zeigt die Dauer einzelner Auswertungen. Die zufällig im Suchraum generierten Parametersätze sind dabei aufsteigend nach benötigter Simulationszeit sortiert. Die einzelnen Auswertungen dauern zwischen 30 und 105 Sekunden, abhängig vom Parametersatz. Diese Zeiten entstammen einem Parcours, welcher der Länge nach in etwa dem späteren Testparcours entspricht.

1.2 Aufbau der Arbeit

Kapitel 2 erläutert Grundlagen der Optimierung (Abschnitt 2.1) und gibt eine Einführung in EAs (Abschnitt 2.2). Dabei liegt die Aufmerksamkeit auf der mehr-kriteriellen evolutionären Optimierung; insbesondere auf dem in dieser Arbeit verwendeten SMS-EMOA (Abschnitt 2.2.4). Zum Abschluss des Kapitels wird in 2.2.5 eine asynchrone Variante des SMS-EMOA vorgestellt.

Das Kapitel 3 befasst sich mit der Anbindung des SMS-EMOA an die Simulation zur Problemlösung. Zu Beginn erklärt Abschnitt 3.1 die genutzten Operatoren. Anschließend analysiert Abschnitt 3.2 die Problemstellung und stellt die zu optimierenden Zielfunktionen vor. Dabei werden außerdem die veränderbaren Parameter und ihr Einfluss auf das Problem erläutert. Es folgt eine Beschreibung der Simulationsumgebung, die zur Optimierung verwendet wird, in Abschnitt 3.3. Abschließend wird das Bindeglied zwischen dem Optimeralgorithmus und der Simulation erklärt und ein Überblick über das Zusammenspiel der eingesetzten Komponenten (SMS-EMOA $\leftrightarrow$ Simulator-Interface $\leftrightarrow$ Simulator) gegeben.

In Kapitel 4 werden die Ergebnisse der Optimierung vorgestellt und auf dem realen Roboter verifiziert. Die erwarteten Ergebnisse werden mit den tatsächlichen in Verbindung gesetzt.

Kapitel 5 zieht ein Fazit der Optimierung und zeigt weitere Analyse und Verbesserungsmöglichkeiten auf.

Kapitel 2

Grundlagen

Dieses Kapitel gibt in Abschnitt 2.1 eine Einführung in die Optimierung. Dabei wird ein besonderes Augenmerk auf die mehrkriterielle Optimierung (Abschnitt 2.1.2) gelegt, da das hier behandelte Problem mehrere Zielfunktionen besitzt. Außerdem gibt es eine Einführung in das Feld der EAs. Der SMS-EMOA als mehrkriterieller evolutionärer Algorithmus wird in Abschnitt 2.2.4 detailliert erklärt, da dieser zur Optimierung verwendet wird.

2.1 Einführung in die Optimierung

Das Ziel der Optimierung ist es, möglichst gute Lösungen im gegebenen Suchraum Ω zu finden. Das Optimierungsproblem ist als eine Funktion $f : \Omega \to \Re$ gegeben, die den Suchraum in einen Lösungsraum abbildet. Diese Funktion wird weiterhin als Zielfunktion oder auch Fitnessfunktion bezeichnet. Sie bewertet die möglichen Lösungen $x \in \Omega$. Je nach Art des Optimierungsproblems wird die Zielfunktion minimiert bzw. maximiert. Bei vielen Problemen ist der Suchraum Ω eine Teilmenge des $\Re^n$. Im Weiteren wird von einer Minimierung der Zielfunktion ausgegangen, da jedes Maximierungsproblem durch Negation der Fitnessfunktion zu einem Minimierungsproblem umgewandelt werden kann [26]:

$$\max(f(x)) = -\min(-f(x))$$

2.1.1 Einkriterielle Optimierung

Die meisten Optimierungsprobleme bestehen allerdings nicht nur aus einer Zielfunktion, sondern beinhalten Nebenbedingungen. Auch diese müssen von der optimalen Lösung erfüllt werden.

2.1.1 Definition (Optimierungsproblem). Ein Optimierungsproblem ist nach [21] definiert als:

$$\begin{aligned} &\text{Minimiere} && f(\boldsymbol{x}), && \boldsymbol{x} \in \Re^n, && (2.1)\\ &\text{unter Bedingung von} && \boldsymbol{g}(\boldsymbol{x}) = 0, && \boldsymbol{g} \in \Re^p, && (2.2)\\ &\text{und} && \boldsymbol{h}(\boldsymbol{x}) \leq 0, && \boldsymbol{h} \in \Re^q. && (2.3) \end{aligned}$$

Der Vektor $\boldsymbol{x}$ enthält die zu optimierenden Entscheidungsvariablen, während p und q die Dimension der Vektorfunktionen $\boldsymbol{g}$ bzw. $\boldsymbol{h}$ sind. Die Variablen p und q geben die Anzahl der Gleichheitsnebenbedingungen (2.2) bzw. Ungleichheitsnebenbedingungen (2.3) an. Für untereinander unabhängige Teilgleichungen $g_i(\boldsymbol{x}) = 0$ mit $i = 1, \ldots, p$ muss $p < n$ gelten. Bei $p = n$ ergibt sich $\boldsymbol{x}$ aus der Lösung des Gleichungssystems (2.2). Für $p > n$ ist das Gleichungssystem 2.2 sogar überbestimmt. Die Anzahl der Ungleichheitsnebenbedingungen q ist hingegen nicht nach oben begrenzt [21].

2.1.2 Mehrkriterielle Optimierung

Bei der mehrkriteriellen Optimierung werden mehrere Anforderungen an eine Problemlösung gestellt. Diese Anforderungen spiegeln sich in verschiedenen Zielfunktionen wider.

2.1.2 Definition (Mehrkriterielles Optimierungsproblem). Ein mehrkriterielles Optimierungsproblem ist nach [8] definiert als:

$$\begin{aligned} &\text{Minimiere} && f_i(\boldsymbol{x}), && i \in \{1, \ldots, d\}, && (2.4)\\ &\text{unter Bedingung von} && g_i(\boldsymbol{x}) = 0, && i \in \{1, \ldots, p\}, &&\\ &\text{und} && h_i(\boldsymbol{x}) \leq 0, && i \in \{1, \ldots, q\}. && \end{aligned}$$

Die Gleichheits- und Ungleichheitsnebenbedingungen bleiben wie in 2.1.1 erhalten. Der Unterschied liegt in der Anzahl der Zielfunktionen d (vgl. 2.4), welche die Dimension des Lösungsraums darstellt. Eine mögliche Lösung $\boldsymbol{x}$ wird dabei von allen Zielfunktionen $f_1(\boldsymbol{x}), \ldots, f_d(\boldsymbol{x})$ bewertet. Das Ziel einer optimalen Lösung ist es, alle Zielfunktionen bestmöglich zu erfüllen. Das Problem dabei ist, dass die Anforderungen typischerweise konfliktär sind und somit nicht von einer Lösung alleine optimal erfüllt werden können.

2.1.3 Beispiel. Ein nachvollziehbares Beispiel aus dem alltäglichen Leben soll konfliktäre Anforderungen veranschaulichen. Die meisten Menschen legen hierzulande größere Strecken mit dem Auto zurück. Bei einer längeren Fahrt mit dem Auto gibt es unterschiedliche Wünsche. Zum einen möchten der Fahrer und seine Gäste möglichst schnell zum Ziel kommen. Auf der anderen Seite möchte, besonders der Fahrer, Sprit sparen. Beide Wünsche sind nicht unbedingt miteinander vereinbar, da bei höherem Tempo zwar das Ziel schneller erreicht, aber auch mehr Sprit verbraucht wird. Daher ist die Lösung des Problems ein Kompromiss aus beiden Wünschen.

Eine weitere Schwierigkeit ist es, verschiedene Lösungen mehrkriterieller Probleme zu vergleichen. Dadurch, dass nicht nur eine Zielfunktion bewertet wird, ist es schwieriger verschiedene Lösungen in Beziehung zueinander zu setzen.

Eine Möglichkeit ist es eine Ersatzzielfunktion $e(\boldsymbol{x})$ zu optimieren, welche die verschiedenen Zielfunktionen kombiniert. Diese Kombination kann z. B. die gewichtete Summe aller d Zielfunktionen sein: $e(\boldsymbol{x}) = \sum_{i=1}^{d} k_i \cdot f_i(\boldsymbol{x})$. In diesem Fall müssen sowohl das Optimierungsproblem als auch die Beziehungen der Zielfunktionen untereinander vor Beginn des Optimierungsprozesses verstanden worden sein. Die Gewichtung der Zielfunktionen führt dazu, dass manche Zielfunktionen einen stärkeren Einfluss auf die Optimierung haben. Diese Präferenzen müssen daher ebenfalls vor Beginn des Optimierungsprozesses bekannt sein [2]. Solche Verfahren werden deshalb als *a priori* Verfahren bezeichnet.

Im Gegensatz dazu stehen *a posteriori* Verfahren. Hier werden die Zielfunktionen nicht zu einer Ersatzzielfunktion aggregiert. Vielmehr wird auf Basis des Funktionsvektors $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_d(\boldsymbol{x}))^\intercal$ direkt optimiert. Dieses Vorgehen entspricht einer echten mehrkriteriellen Optimierung. Der Vorteil ist, dass vor Beginn des Optimierungsprozesses keine Präferenz angegeben werden muss. Dadurch ist kein gesondertes Wissen über den Zusammenhang der Zielfunktionen notwendig. Die Priorisierung der Zielfunktionen wird, wie der Name schon andeutet, erst im Anschluss an den Optimierungsprozess festgelegt. An dieser Stelle liegt eine Lösungsmenge vor, aus der ersichtlich ist, welche Anforderungen in welcher Weise erfüllt werden können und welche Kompromisse mit anderen Anforderungen möglich sind. Wir befinden uns dabei im Bereich der sogenannten Vektor- bzw. *Pareto-Optimierung* [2].

Der Vergleich einzelner Lösungen ist nicht mehr trivial, da die Entscheidung, welche Lösung die bessere ist, nicht nur an einer Zielfunktion festgemacht wird. Klar ist, dass Lösung $\boldsymbol{x_1}$ die Lösung $\boldsymbol{x_2}$ übertrifft, wenn sie in mindestens einer Zielfunktion besser und in keiner anderen schlechter ist als $\boldsymbol{x_2}$. Um die Relationen genauer betrachten zu können, folgen einige Definitionen.

2.1.4 Definition (Dominanz(-relationen)). Für zwei n-dimensionale Vektoren von Entscheidungsvariablen $\boldsymbol{x}_1$ und $\boldsymbol{x}_2$ und ihren d-dimensionalen Zielfunktionsvektoren $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_d(\boldsymbol{x}))$ gilt:

Eine Lösung $\boldsymbol{x}_1$ dominiert eine andere Lösung $\boldsymbol{x}_2$ ($\boldsymbol{x}_1 \prec \boldsymbol{x}_2$), wenn:

$$\exists j \in \{1, \ldots, d\} : f_j(\boldsymbol{x}_1) < f_j(\boldsymbol{x}_2) \ \wedge \ \forall i \in \{1, \ldots, d\} \setminus j : f_i(\boldsymbol{x}_1) \leq f_i(\boldsymbol{x}_2)$$

Eine Lösung $\boldsymbol{x}_1$ dominiert eine andere Lösung $\boldsymbol{x}_2$ schwach ($\boldsymbol{x}_1 \preceq \boldsymbol{x}_2$), wenn:

$$\forall i \in \{1, \ldots, d\} : f_i(\boldsymbol{x}_1) \leq f_i(\boldsymbol{x}_2)$$

Die Lösungen $\boldsymbol{x}_1$ und $\boldsymbol{x}_2$ sind unvergleichbar ($\boldsymbol{x}_1 || \boldsymbol{x}_2$), wenn:

$$(\boldsymbol{x}_1 \not\preceq \boldsymbol{x}_2) \ \wedge \ (\boldsymbol{x}_2 \not\preceq \boldsymbol{x}_1)$$

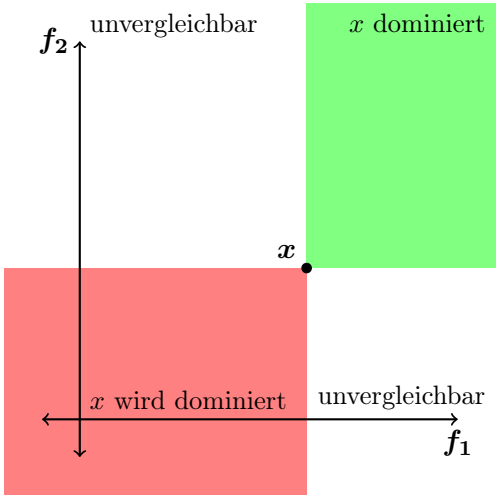


Abbildung 2.1: Die Relation der Lösung $\boldsymbol{x}$ zu anderen Lösungen im 2-dimensionalen Lösungsraum

Die Dominanzrelation der Lösung $\boldsymbol{x}$ zu anderen Lösungen ist in Abbildung 2.1 für einen zweidimensionalen Lösungsraum dargestellt. Die Abbildung zeigt, dass die Wahrscheinlichkeit, dass zwei Lösungen vergleichbar sind, in diesem Fall bei $\frac{1}{2}$ liegt. Diese Wahrscheinlichkeit sinkt mit zunehmender Dimension des Lösungsraums d. Sie ist bestimmt durch $(\frac{1}{2})^{d-1}$.

2.1.5 Definition (nicht-Dominanz und Pareto-Optimalität). Die Lösung $\boldsymbol{x_1} \in M$ gilt als nicht dominiert, wenn es keine Lösung $\boldsymbol{x_2} \in M$ gibt mit $\boldsymbol{x}_2 \preceq \boldsymbol{x}_1$.

$$\nexists \boldsymbol{x}_2 \in M : \boldsymbol{x}_2 \prec \boldsymbol{x}_1$$

Eine Lösung $\boldsymbol{x}_1$ ist Pareto-optimal, wenn M den gesamten Suchraum abdeckt ($M = \Omega$).

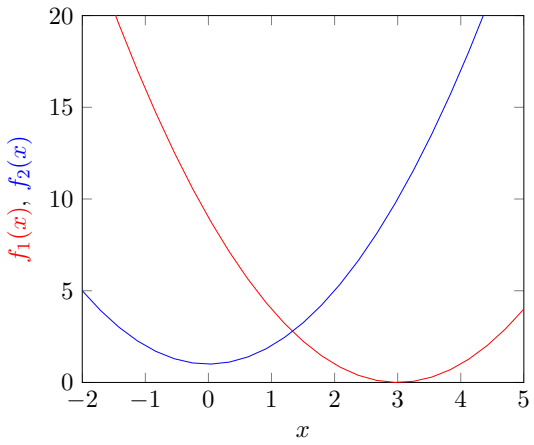
2.1.6 Definition (nicht-dominierte Menge, Pareto-Menge, Pareto-Front). Enthält eine Menge M alle nicht-dominierten Lösungen $\boldsymbol{x} \in M$, so gilt sie als *nicht-dominierte Menge* $ND(M)$.

$$ND(M) = \{\boldsymbol{x}_1 \in M \mid \nexists \boldsymbol{x}_2 : \boldsymbol{x}_2 \prec \boldsymbol{x}_1\}$$

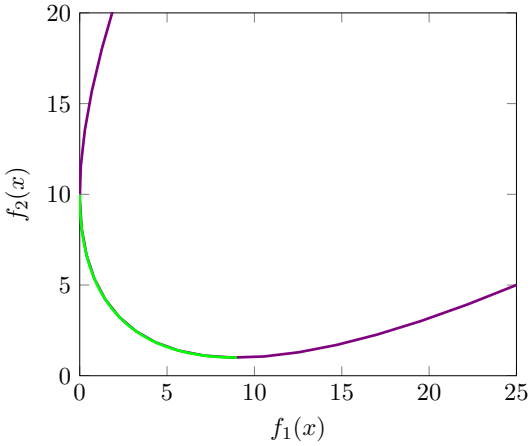
Die Elemente einer nicht-dominierten Menge $ND(M)$ sind untereinander unvergleichbar oder dominieren einander schwach. Eine nicht-dominierte Menge $ND(M)$ heißt *Pareto-Menge* PM, wenn sich M über den gesamten Suchraum erstreckt ($M = \Omega$). Die Menge der Zielfunktionsvektoren zu den Lösungen der Pareto-Menge $\boldsymbol{x} \in PM$ wird als *Pareto-Front* bezeichnet [5].

$$PF = \{\boldsymbol{f} = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))^\intercal \mid \boldsymbol{x} \in PM\}$$

In der Praxis ist die Pareto-Menge schwer zu erreichen. Besonders in kontinuierlichen Suchräumen enthält PM unendlich viele Elemente. Deshalb wird eine endliche Menge von Lösungen gesucht, welche die unbekannte Pareto-Front möglichst gut approximiert.



(a) Die Zielfunktionen $f_1(x)$ und $f_2(x)$ im eindimensionalen Suchraum.



(b) Die Zielfunktionen $f_1(x)$ und $f_2(x)$ im zweidimensionalen Lösungsraum. Die Pareto-Front ist grün hervorgehoben.

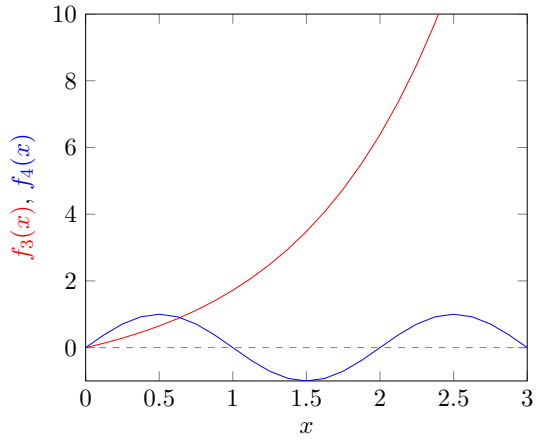
Abbildung 2.2: Abbildung des eindimensionalen Suchraums (2.2a) in den zweidimensionalen Lösungsraum (2.2b).

Dazu gehört, dass die Punkte der Lösungsmenge möglichst nah an *PF* liegen und sich, idealerweise gleichmäßig, über die gesamte *Pareto-Front* verteilen.

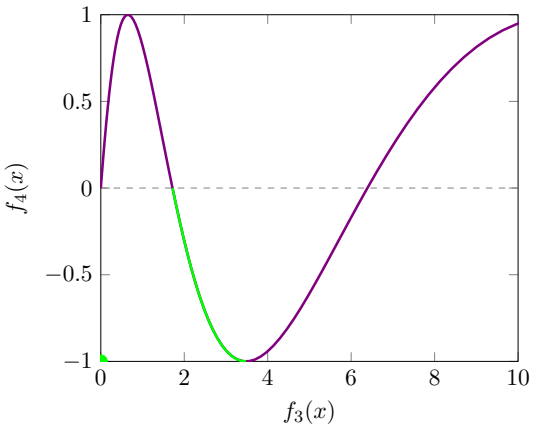
Für das Beispiel 2.1.3 bedeutet es, dass bei der *a priori* Methode zuvor eine Priorisierung der Fahrzeit und des Spritverbrauchs erfolgen muss. Beim *a posteriori* Verfahren hingegen kann aus der *Pareto-Menge* möglicher Kombinationen aus Fahrzeit und Spritverbrauch gewählt werden.

2.1.7 Beispiel. Zur Veranschaulichung der Pareto-Front werden zwei Zielfunktionen $f_1(x) = (x-3)^2$ und $f_2(x) = x^2 + 1$ betrachtet. Beide sind von einer Entscheidungsvariablen $x \in \Re$ abhängig. Der Verlauf der Zielfunktionen im Suchraum ist in Abbildung 2.2a aufgezeichnet. Die Pareto-Menge des durch $f_1(x)$ und $f_2(x)$ beschriebenen Optimierungsproblems ist $PM = \{x \mid x \in [0,3]\}$. Der Lösungsraum der beiden Funktionen ist in Abbildung 2.2b veranschaulicht. Die Pareto-Front ist dabei grün kenntlich gemacht worden.

2.1.8 Beispiel. Ein weiteres Optimierungsproblem hängt von einer Entscheidungsvariablen $x \in \Re^+$ ab. Es ist durch die Zielfunktionen $f_3(x) = e^x - 1$ und $f_4(x) = \sin(180 \cdot x)$ beschrieben. Der Verlauf der Zielfunktionen im Suchraum ist in Abbildung 2.3a aufgezeichnet. Die Pareto-Menge des Problems ist $PM = \{x \mid x \in [0, 0.5] \vee x \in [1, 1.5]\}$. Der Lösungsraum der beiden Funktionen ist in Abbildung 2.3b veranschaulicht. Die Pareto-Front ist wie in Abbildung 2.2b grün kenntlich gemacht worden. Dieses zweite Problem zeigt, dass die Pareto-Front nicht zwingend zusammenhängend ist.



(a) Die Zielfunktionen $f_3(x)$ und $f_4(x)$ im eindimensionalen Suchraum.



(b) Die Zielfunktionen $f_3(x)$ und $f_4(x)$ im zweidimensionalen Lösungsraum. Die Pareto-Front ist grün hervorgehoben.

Abbildung 2.3: Abbildung des eindimensionalen Suchraums (2.3a) in den zweidimensionalen Lösungsraum (2.3b).

2.2 Evolutionäre Algorithmen

2.2.1 Inspiration durch Evolution

Evolutionäre Algorithmen (EAs) sind randomisierte Suchheuristiken, die zur Optimierung komplexer Probleme eingesetzt werden. Sie sind an Darwins Vorstellung der biologischen Evolution angelehnt. Die Idee ist, dass sich hochkomplexe Lebensformen durch Veränderung ihres Erbgutes immer besser an ihre Umwelt anpassen. Eine Veränderung der Lebensformen erfolgt durch eine Kombination aus Reproduktion, Variation und natürlicher Auslese. Dabei kann die fortwährend verbesserte Anpassung der Lebensformen an ihre Umwelt als Optimierungsproblem interpretiert werden. Das Optimum im Suchraum entspricht in diesem Fall der bestmöglichen Anpassung an die Umwelt. EAs versuchen diese durch randomisiert gesteuerte Veränderungen der Variablen (des Phänotyps) zu erreichen. Ein Individuum enthält neben einem Satz von Entscheidungsvariablen weiterhin einen Zielfunktionsvektor, der die Basis für die spätere Selektion bildet.

Die Ursprünge der heutzutage unter dem Oberbegriff Evolutionäre Algorithmen (EAs) zusammengefassten Verfahren sind unterschiedlich. Zum einen gibt es die von Schwefel und Rechenberg entwickelten *Evolutionsstrategien* [22], zum anderen die *genetischen Algorithmen* [18], die von Holland entwickelt wurden. Weitere Verfahren sind das *evolutionäre Programmieren (EP)* von Fogel [15] sowie die *genetische Programmierung (GP)* von Koza [20]. Die Ansätze unterscheiden sich durch die verschiedenen Anwendungsgebiete, für die sie entwickelt wurden. So nutzten die ersten genetischen Algorithmen stets eine binäre Kodierung des Suchraums, wohingegen Evolutionsstrategien eine problemspezifische, meist

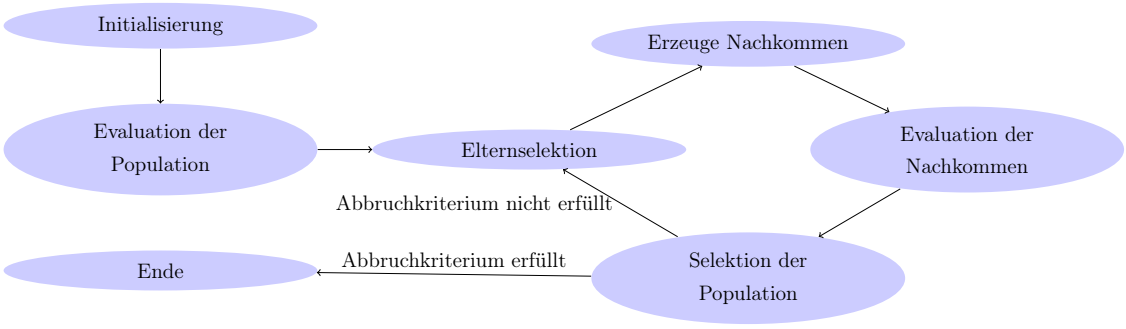


Abbildung 2.4: Schematische Darstellung eines EAs

reellwertige, Repräsentation verwendeten. Mittlerweile sind die Grenzen zwischen den einzelnen Konzepten durch ihren gegenseitigen Einfluss verschwommen, und es werden alle Ansätze unter dem Begriff evolutionäre Algorithmen zusammengefasst.

2.2.2 Algorithmus

Der Hauptteil des Optimierprozesses läuft beim EA iterativ ab. Einzelne Iterationen werden als *Generationen* bezeichnet. Die Individuen der neusten Generation bilden die aktuelle *Population*. Aus dieser werden zunächst Elter-Individuen gewählt auf deren Basis Nachkommen erzeugt werden. Die Nachkommen werden mittels Rekombination und/oder Mutation erzeugt bzw. verändert. Bei der *Rekombination* werden die Genome der Elter-Individuen gemischt, während bei der *Mutation* eine Veränderung des Genoms stattfindet. Das resultierende Individuum wird anschließend durch Auswertung der Zielfunktion(en) bewertet. Im Anschluss konkurrieren die Individuen um die Aufnahme in die nächste Generation. Dabei gilt das Prinzip *survival-of-the-fittest*. Die bestbewertetsten Individuen schaffen den Sprung in die nächste Generation. In Abhängigkeit vom Abbruchkriterium wird die Schleife entweder verlassen und der Algorithmus terminiert oder der Optimierprozess startet mit der Auswahl der Elter-Individuen erneut. Bevor der rundenbasierte Hauptteil mit der Elternselektion beginnt, wird der EA initialisiert und die initiale Population bewertet. Diesen Ablauf verdeutlicht Abbildung 2.4.

Initialisierung Der Algorithmus startet mit der Initialisierung der Strategieparameter und der ersten Population. Die Individuen werden meist aus zufällig gleichverteilten Lösungen im Suchraum generiert. Es besteht außerdem die Möglichkeit die initiale Population mit bereits bekannten Lösungen, die es weiter zu verbessern gilt, zu initialisieren. Dieser Fall wird als *Warmstart* bezeichnet. Zudem werden Strategieparameter wie Populationsgröße μ und die Anzahl ihrer Nachkommen λ initialisiert. Abhängig vom gewählten Algorithmus gibt es weitere spezifische Strategieparameter.

Evaluation (der Population) Bewertet die μ Individuen der initialen Population anhand der Zielfunktion(en). Meist werden die Individuen direkt nach ihrer Generie-

rung bewertet. Die Auswertung eines Individuums kann sehr zeitaufwändig sein, da evtl. eine zeitintensive Simulation notwendig ist.

Elternselektion Bei der Elternselektion werden Elter-Individuen ausgewählt, aus denen Nachkommen erzeugt werden. Die Auswahl passiert meistens basierend auf den Zielfunktionswerten der Individuen oder zufällig gleichverteilt über die Population.

Variation (Erzeugen von Nachkommen) Die Variation generiert durch Veränderung der Genome der Elter-Individuen verschiedenartige Nachkommen. Dies dient der Exploration des Suchraums. Die Variation besteht aus Rekombination und/oder Mutation. Während bei der Rekombination typischerweise zwei (teilweise auch mehr) Elter-Individuen involviert sind, variiert die Mutation die Genome nur eines Individuums.

Rekombination Bei der Rekombination werden Genome der Elter-Individuen vermischt. Zur Rekombination gibt es verschiedene Operatoren, die abhängig von der gewählten Repräsentation der Daten (Genome) sind.

Mutation Die Mutation dient dazu, ein Individuum zufallsgesteuert leicht zu verändern. Im Falle einer vorherigen Rekombination wird sie auf das dort erzeugte Individuum angewendet. Wird zuvor keine Rekombination durchgeführt, findet sie auf einem geklonten (kopierten) Elter-Individuum statt. Für reellwertige Suchräume werden meist normalverteilte Zufallszahlen addiert. In vielen Algorithmen, welche diese Mutation verwenden, gibt es einen zusätzlichen Strategieparameter, der die Größe der Veränderung beeinflusst. Er wird meist als Schrittweite bezeichnet. Bei mehreren Entscheidungsvariablen kann es Schrittweiten für jede einzelne Variable geben.

Evaluation (der Nachkommen) Bewertet die λ Nachkommen anhand der Zielfunktion(en). Die Auswertung eines Individuums kann sehr zeitaufwändig sein, da evtl. eine zeitintensive Simulation notwendig ist.

Selektion (der Population) Bei der Selektion werden die Individuen für die nächste Generation ausgewählt. Es gibt zwei Basis-Strategien zur Selektion; die Komma- und die Plus-Selektion. Die (μ, λ)-Strategie selektiert aus λ Nachkommen die besten μ Individuen für die Folgepopulation. In diesem Fall muss $\mu \leq \lambda$ gelten, wobei $\mu = \lambda$ einem „random walk“ mit ggf. mehreren Wegen entspricht. Es ist zu beachten, dass die Selektion ausschließlich aus den erzeugten Nachkommen generiert wird. Dadurch werden Individuen der aktuellen Population nicht in die nächste Generation aufgenommen, selbst wenn sie bessere Zielfunktionswerte haben. Bei der $(\mu + \lambda)$-Strategie hingegen wird aus allen Individuen selektiert. Das meint sowohl die λ Nachkommen als auch die μ Individuen der aktuellen Population. In diesem Fall bleibt ein Individuum im Optimierprozess bis es μ bessere Individuen gibt. Allerdings besitzt die

Plus-Selektion gegenüber der Komma-Selektion auch einen entscheidenden Nachteil: Es besteht immer die Gefahr lange in einem lokalen Minimum zu verharren oder dieses überhaupt nicht zu verlassen.

Ansätze, die versuchen die positiven Eigenschaften beider Selektionsstrategien zu vereinen, wählen z. B. die Plus-Selektion und geben eine maximale Lebensdauer $0 \leq \kappa \leq \infty$ für Individuen an. Hat ein Individuum diese Anzahl an Generationen überlebt, wird es aus der Selektion ausgeschlossen, ungeachtet davon, ob es zu den μ besten Individuen gehört. Für $\kappa = \infty$ kann ein Individuum nur durch bessere Individuen aus der Population verdrängt werden.

Abbruchkriterium Ein Abbruchkriterium gibt an, unter welchen Umständen der Optimierprozess beendet werden soll. Ein Optimum ist in der Regel nicht zu erreichen bzw. schwer als solches zu verifizieren. Das Abbruchkriterium beschränkt typischerweise Ressourcen (Laufzeit, Anzahl von Generationen) oder bezieht sich auf die Qualitätssteigerung der Lösungen zwischen den Generationen. Bezieht sich ein Abbruchkriterium z. B. auf die Schrittweite der Mutation, wird der Optimierprozess gestoppt, wenn eine zuvor festgelegte, minimale Schrittweite erreicht wurde und man davon ausgehen kann, dass keine signifikante Verbesserung mehr erzielt wird.

Einzelne EAs unterscheiden sich vor allem durch ihre verwendeten Operatoren, sowohl für die Rekombination und Mutation als auch für die Selektion. Über sie kann Wissen über das Problem in den EA einfließen.

2.2.3 Evolutionäre Mehrkriterielle Optimierung

Evolutionäre Mehrkriterielle Optimierung ist ein noch recht junger Forschungsbereich. Als erster *Evolutionärer Mehrkriterieller Optimier-Algorithmus (EMOA)* gilt der 1985 in [23] vorgestellte *Vector Evaluated Genetic Algorithm (VEGA)* von Schaffer. Goldberg brachte mit dem *Non-Dominated Sorting (NDS)* den Forschungsbereich ein großes Stück voran [17]. Dabei wird die Population in Mengen von nicht-dominierten Lösungen unterteilt. Für jede Teilmenge TM gilt: $\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in TM \ : \ \boldsymbol{x}_1 || \boldsymbol{x}_2$. Die erste Menge nicht-dominierter Lösungen der Population bildet die erste Front. Den Individuen dieser Front wird der Rang 1 zugewiesen. Der Vorgang wird mit den restlichen Individuen der Population zur Erstellung der zweiten Front (Individuen Rang 2) wiederholt. Sind alle Individuen einer Front zugeordnet, ist der Sortierprozess abgeschlossen. Das Erstellen einer solchen Sortierung nimmt $O(MN^3)$ Laufzeit in Anspruch. Dabei ist M die Dimension des Lösungsraums/Anzahl der Zielfunktionen und N die Anzahl der Individuen. Die Laufzeit des NDS konnte, wie in [27] beschrieben, verbessert werden.

Die Schwierigkeit der evolutionären mehrkriteriellen Optimierung ist die Selektion der Individuen für die Folgepopulation. Der NSGAII als Nachfolger des ersten NDS nutzenden

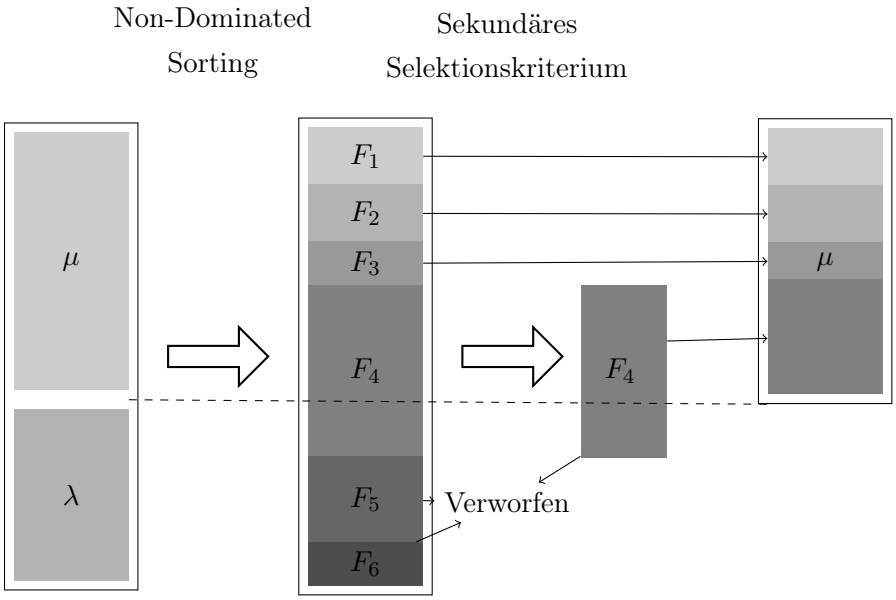


Abbildung 2.5: Diese Abbildung zeigt, wie in zwei Schritten aus $(\mu + \lambda)$ die μ besten Individuen der neuen Population ausgewählt werden. Zunächst trifft Non-Dominated Sorting eine Vorauswahl. Die Fronten F_1, F_2 und F_3 werden direkt in die nächste Generation aufgenommen. Dadurch fehlt Platz, um alle Individuen aus F_4 aufzunehmen. An dieser Stelle greift das sekundäre Selektionskriterium - *S-Metrik (Hypervolumen)* beim SMS-EMOA bzw. *crowding-distance* beim NSGAII. Die zu F_4 gehörenden Individuen werden anhand des sekundären Selektionskriteriums bewertet: Die Besten werden in die neue Population aufgenommen, die Übrigen werden zusammen mit den Individuen der Fronten F_5 und F_6 verworfen und scheiden aus dem Optimierprozess aus. Diese Grafik ist an Abbildung 2 aus [11] angelehnt.

Algorithmus, *Nondominated Sorting Genetic Algorithm (NSGA)*, erweitert diesen um eine *elitäre* Selektionsstrategie und reduziert die Laufzeit des NDS ($O(MN^2)$). Hierbei meint *elitär*, dass garantiert die besten μ Individuen, unter Gesichtspunkten der mehrkriteriellen Optimierung (vgl. Abschnitt 2.1.2), in die Folgepopulation aufgenommen werden. Dazu ist es notwendig, dass das sekundäre Selektionskriterium eine Funktion $f : \Re^n \mapsto \Re$ ist, um eindeutig die besten Individuen zu identifizieren. Als primäres Selektionskriterium wird weiterhin NDS verwendet, allerdings mit verbesserter Laufzeit. Als sekundäres Selektionskriterium wird die *crowding-distance* eingeführt [11]. Sollten nach der Auswertung durch das primäre Selektionskriterium mehr als μ Individuen diese Auslese überstehen, werden die überschüssigen Individuen mittels des sekundären Selektionskriteriums bestimmt (vgl. Abb. 2.5). Der NSGAII nutzt die *crowding-distance* als sekundäres Selektionskriterium. Diese fördert die gleichmäßige Abdeckung der Pareto-Front.

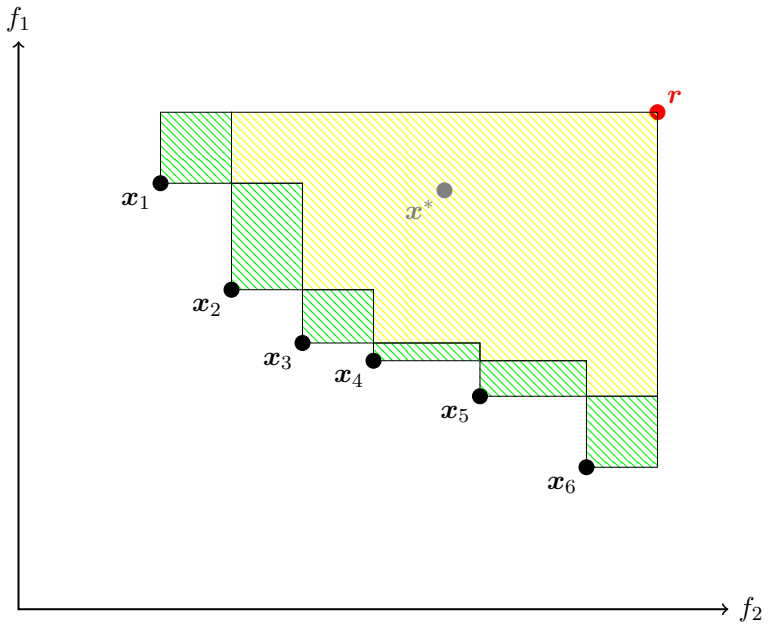


Abbildung 2.6: Dieses Beispiel veranschaulicht ein zweidimensionales Minimierungsproblem. Die Abbildung zeigt das dominierte Hypervolumen der nicht-dominierten Lösungen $\boldsymbol{x}_1, \ldots \boldsymbol{x}_6$ mit dem Referenzpunkt $\boldsymbol{r}$ (Gesamtfläche, gelb und grün). Die grünen Flächen kennzeichnen den exklusiven Beitrag einer jeden Lösung zum Gesamtvolumen. Wie in der Abbildung zu sehen ist, leistet der dominierte Punkt $\boldsymbol{x}^*$ keinen Beitrag zum dominierten Hypervolumen.

2.2.4 S-Metrik-Selektion-EMOA

Der in dieser Arbeit verwendete *S-Metrik-Selektion-EMOA (SMS-EMOA)* aus Emmerich et al. [14] nutzt das dominierte Hypervolumen, auch bekannt als *S-Metrik*, als sekundäres Selektionskriterium. Das primäre Selektionskriterium bleibt das *Non-Dominated Sorting (NDS)*. Sein großer Vorteil gegenüber dem NSGAII ist, dass er auch in hochdimensionalen Lösungsräumen funktioniert [3].

S-Metrik

Die *S-Metrik* wurde ursprünglich als Qualitätsindikator einer Pareto-Front formuliert [30]. Sie berechnet die Größe des Raumes, der von einer Menge nicht-dominierter Lösungen mit einem Referenzpunkt überdeckt/dominiert wird. Aus mathematischer Sicht entspricht dies dem Lebesgue-Maß [26]. Beim SMS-EMOA wird sie dazu verwendet, den exklusiv dominierten Anteil eines Individuums am Gesamtvolumen zu berechnen (vgl. Abb. 2.6).

2.2.1 Definition (S-Metrik). Der Wert der S-Metrik $S(M, \boldsymbol{r})$ entspricht der Größe des Raums, der von den Punkten der Menge M überdeckt wird. Der Raum ist dabei durch den Referenzpunkt $\boldsymbol{r}$ begrenzt. Λ ist das Lebesgue-Maß [26].

$$S(M, \boldsymbol{r}) = \Lambda\left(\left\{\bigcup_{m \in M} \{\boldsymbol{x} | \boldsymbol{m} \preceq \boldsymbol{x} \preceq \boldsymbol{r}\}\right\}\right)$$

Der Wert des exklusiv dominierten Hypervolumens wird beim SMS-EMOA für die Selektion genutzt. Das Individuum, das den geringsten Beitrag am Gesamtvolumen leistet, wird verworfen. Dadurch steigt der Wert des dominierten Hypervolumens monoton.

Die Berechnung der S-Metrik ist sehr zeitintensiv ($O(n^{d-1})$); n kennzeichnet die Anzahl der involvierten Individuen im d-dimensionalen Lösungsraum. Beume und Rudolph senkten die Laufzeit auf $O(n \log n + n^{d/2})$, indem sie die Berechnung auf einen speziellen Fall von *Klee's measure problem* reduzierten [4].

Algorithmus

Der SMS-EMOA ist ein $(\mu + 1)$-EA. Jede Generation erzeugt nur ein Individuum. Das hat den Vorteil, dass für die Folgepopulation auch nur ein Individuum verworfen werden muss. Dies ist das Individuum mit dem niedrigsten S-Metrik-Wert aus der schlechtesten Front. Übertragen auf Abbildung 2.5 ist $\lambda = 1$. Damit ist die kritische Front (in Abbildung 2.5 Front F_4) immer die schlechteste Front.

Algorithmus 2.1 stellt den Ablauf des SMS-EMOA dar. Die μ Individuen der initialen Population werden entweder mit bekannten Lösungen oder zufällig initialisiert. Die Methode *createOffspring* generiert in jedem Schleifendurchlauf (Generationswechsel) zunächst einen Nachkommen. Dazu können problemspezifische Operatoren genutzt werden. In den meisten Fällen werden aus der aktuellen Population P_{count} zwei Individuen zufällig ausgewählt, miteinander rekombiniert und der Nachkomme wird mutiert. Das vom NSGAII übernommene *fastNonDominatedSort* unterteilt die temporäre Menge $Q_{count} = P_{count} \cup \{i\}$ in nicht-dominierte Fronten $(F_1, \ldots, F_w)$. Anschließend wird der zur S-Metrik Berechnung benötigte Referenzpunkt $\boldsymbol{r}$ konstruiert. Dabei wird die schlechteste Komponente aller Individuen in F_w in jeder Dimension gesucht und um einen offset verschoben. Für $\boldsymbol{r}$ gilt $\forall \boldsymbol{x} \in F_w : \boldsymbol{x} \prec \boldsymbol{r}$. Mit Hilfe dieses Referenzpunktes wird der Hypervolumenbeitrag $\Delta_s(\boldsymbol{x}, F_w, \boldsymbol{r})$ jedes Individuums $\boldsymbol{x}$ in F_w berechnet.

$$\Delta_s(\boldsymbol{x}, F_w, \boldsymbol{r}) = S(F_w, \boldsymbol{r}) - S(F_w \setminus \{\boldsymbol{x}\}, \boldsymbol{r})$$

Die nächste Generation $P_{count+1}$ wird aus der temporären Menge Q_{count} ohne dem Individuum mit geringstem Hypervolumenbeitrag $\boldsymbol{x}^*$ berechnet.

Algorithmus 2.1 SMS-EMOA

1: $P_0 \leftarrow \text{init}()$ // Initialisiere erste Population aus μ Individuen
2: $count \leftarrow 0$ // Generations-Zähler
3: **while** Abbruchkriterium nicht erfüllt **do**
4: $\boldsymbol{i} \leftarrow \text{createOffspring}(P_{count})$ // Erzeuge 1 Nachkommen
5: $\text{evaluate}(i)$ // Berechne Zielfunktionswerte des Nachkommen
6: $Q_{count} \leftarrow P_{count} \cup \{i\}$
7: $\{F_1, \ldots, F_w\} \leftarrow \text{fastNondominatedSort}(Q_{count})$ // Sortiere in w Fronten
8: $\boldsymbol{r} \leftarrow \text{createReferencePoint}(F_w)$ // Berechne Referenzpunkt zur Front F_w
9: $\boldsymbol{x}^* \leftarrow \text{argmin}_{\boldsymbol{x} \in F_2}(\Delta_s(\boldsymbol{x}, F_w, \boldsymbol{r}))$ // Bestimmte $\boldsymbol{x}^*$ mit kleinstem $\Delta_s(\boldsymbol{x}, F_w, \boldsymbol{r})$
10: $count \leftarrow count + 1$
11: $P_{count} \leftarrow Q \setminus \{\boldsymbol{x}^*\}$ // Entferne schlechtestes Element und setze nächste Generation
12: **end while**

2.2.5 Asynchroner S-Metrik-Selektion-EMOA

Diese Arbeit lässt zur Problemlösung eine parallele Auswertung der Individuen zu, daher ist der SMS-EMOA auch als asynchroner Algorithmus implementiert. Die Selektion bleibt dabei eine $(\mu + 1)$-Strategie. Eine (synchronisierte) Parallelisierung der Auswertung durch Anhebung der Anzahl von Nachkommen, würde zu einem erhöhten Aufwand der S-Metrik führen. Große Differenzen in der Evaluationszeit einzelner Individuen würden in Wartezeit auf die zeitintensivste Auswertung resultieren. Denn erst nach der Bewertung aller Nachkommen können neue Individuen generiert werden. Diese Leerlaufzeiten verschwenden Ressourcen [19]. Abbildung 2.7 zeigt ein beispielhaftes Verhältnis zwischen Arbeits- und Leerlaufzeit. Die Zeit, in der ein Prozess arbeitet/einen Funktionswert berechnet, ist durch eine horizontale schwarze Linie dargestellt; gestrichelte horizontale Linien sind Leerlaufzeiten der Prozesse. Dieses Beispiel zeigt die Auswertung von vier Generationen eines parallelisierten $(\mu + 6)$-EA. Die sechs Nachkommen werden von den Prozessen parallel ausgewertet. Eine Auswertung dauert zwischen einer und fünf Zeiteinheiten. Die durchgezogenen vertikalen Linien kennzeichnen die Wechsel von einer Generation zur nächsten. An dieser Stelle müssen die einzelnen Auswertungsprozesse miteinander synchronisiert werden; dort entsteht der Zeitverlust. Die Abbildung veranschaulicht, dass 31 der insgesamt 84 Zeitschritte (6 Prozesse zu je 14 Zeitschritten) ungenutzt bleiben. Die Prozesse verbringen damit knapp 37% ihrer Zeit im Leerlauf.

In Zukunft soll diese Leerlaufzeit minimiert werden bzw. verschwinden. Die Idee ist, dass mehrere Prozesse ihre gemeinsame Population verbessern. Das Selektionsschema des asynchronen SMS-EMOA bleibt ein $(\mu + 1)$ Schema. Sobald ein Prozess die Auswertung abgeschlossen hat, bindet er das bewertete Individuum direkt in die aktuelle Population ein. Einbinden kann in diesem Fall auch bedeuten, es zu verwerfen, falls das Individu-

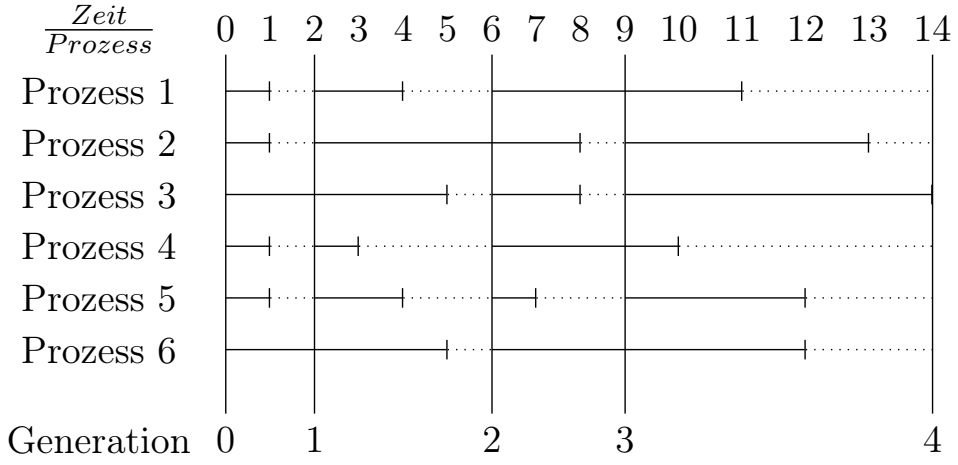


Abbildung 2.7: Vergleich von Leerlaufzeit zu Arbeitszeit der verschiedenen Prozesse

um keine Verbesserung bringt. Sollte das Abbruchkriterium noch nicht erfüllt sein, wird anschließend das nächste Individuum aus der aktualisierten Population generiert und wiederum bewertet. Jeder einzelne Prozess führt somit alle Aufgaben eines SMS-EMOA aus (vgl. Algorithmus 2.2) - nicht nur die Evaluation - wie im parallelen Fall.

Algorithmus 2.2 Hauptteil des asynchronen SMS-EMOA

1: $P_0 \leftarrow \text{init}(mutex)$
2: **while** Abbruchkriterium nicht erfüllt **do**
3: $\boldsymbol{i} \leftarrow \text{createOffspring}(P_{count},\ mutex)$ // Erzeuge 1 Nachkommen
4: $\text{evaluate}(\boldsymbol{i})$ // Berechne Zielfunktionswerte des Nachkommen
5: $\text{enter}(mutex)$ // Verriegle Population für andere Prozesse
6: $Q_{count} \leftarrow P_{count} \cup \{\boldsymbol{i}\}$
7: $\{F_1, \ldots, F_w\} \leftarrow \text{fastNondominatedSort}(Q_{count})$ // Sortiere in w Fronten
8: $\boldsymbol{r} \leftarrow \text{createReferencePoint}(F_w)$ // Berechne Referenzpunkt zur Front F_w
9: $\boldsymbol{x}^* \leftarrow \text{argmin}_{\boldsymbol{x} \in F_2}(\Delta_s(\boldsymbol{x}, F_w, \boldsymbol{r}))$ // Bestimmte $\boldsymbol{x}^*$ mit kleinstem $\Delta_s(\boldsymbol{x}, F_w, \boldsymbol{r})$
10: $count \leftarrow count + 1$
11: $P_{count} \leftarrow Q \setminus \{\boldsymbol{x}^*\}$ // Entferne schlechtestes Element und setze nächste Generation
12: $\text{leave}(mutex)$ // Gebe Population für andere Prozesse frei
13: **end while**

Algorithmus 2.3 beschreibt die globale Initialisierung des asynchronen SMS-EMOA. Er legt die geteilte Population mit Speicherplatz für μ Individuen, die einzelnen Prozesse und ein Mutex zu deren Synchronisation an. Außerdem initialisiert er den Generationszähler und startet die asynchrone Ausführung. Jeder Prozess führt den Code aus Algorithmus 2.2 aus. Die Population könnte auch vor dem Start der einzelnen Prozesse mit μ Individuen

initialisiert werden, dabei käme allerdings der Vorteil der asynchronen Evaluation bei der initialen Population nicht zum Tragen.

Algorithmus 2.3 Initialisierung des asynchronen SMS-EMOA

```
1: P ← init(μ)                    // Initialisiere Platzhalter für μ Individuen
2: tp ← initThreadPool(τ)         // Initialisiere Threadpool mit τ Threads
3: count ← 0                      // Generations-Zähler
4: mutex ← initMutex()            // Initialisiere Mutex zur Synchronisation
5: start(tp)                      // Starte asynchrone Ausführung des SMS-EMOA
```

Die Prozesse greifen während der Selektion der Elter-Individuen ebenfalls auf die Population zu. Deshalb muss die Methode *createOffspring* dafür sorgen, dass die Population währenddessen nicht verändert wird (vgl. Algorithmus 2.4).

Algorithmus 2.4 Die Funktion *createOffsrping* des asynchronen SMS-EMOA

Eingabe: population, mutex

```
1: enter(mutex)                   // Verriegle Population für andere Prozesse
2: parents ← select(population)   // Selektiert die Elter-Individuen
3: leave(mutex)                   // Gebe Population für andere Prozesse frei
4: child ← crossover(parents)     // Rekombination
5: child ← mutate(child)          // Mutation
```

Ausgabe: child

Die Abbildung 2.8 zeigt zum Vergleich den Bewertungsverlauf des asynchronen SMS-EMOA unter gleichen Bedingungen wie bei der parallelen Auswertung in Abbildung 2.7. Gleiche Bedingungen beziehen sich dabei auf die gleiche Anzahl von Prozessen, Auswertungen und deren Abfolge sowie Dauer. Wie in der Abbildung zu sehen ist, gibt es keine Leerlaufzeiten mehr. Außerdem ist der Optimierprozess über vier Generationen drei Zeitschritte früher beendet, welches einer prozentualen Ersparnis von circa 22% entspricht. Bei nur vier Generationswechseln (Iterationen) fällt die Dauer der letzten Auswertungen zu stark ins Gewicht, als dass eine bessere Zeitersparnis erreicht werden kann. Je mehr Iterationen durchgeführt werden, desto eher wird die maximal mögliche Ersparnis erreicht.

Aus Abbildung 2.8 geht weiterhin hervor: Zum diskreten Zeitpunkt 5 sind vier Bewertungen abgeschlossen. Die Prozesse 1, 2, 5 und 6 versuchen gleichzeitig ihr Individuum in die Population einzugliedern, weshalb der Zugriff auf die Population synchronisiert werden muss. In der Implementierung, die im Zuge dieser Arbeit vorgenommen wurde, darf nur ein Prozess auf der Population arbeiten. Dadurch entstehen während der Aktualisierung der Population und dem Erzeugen von Nachkommen Leerlaufzeiten. Diese beinhalten auch die Berechnung der S-Metrik. Bei starken Differenzen in der Dauer einer Evaluation fallen diese Wartezeiten, falls sie überhaupt auftreten, nicht ins Gewicht; die Vorteile überwiegen.

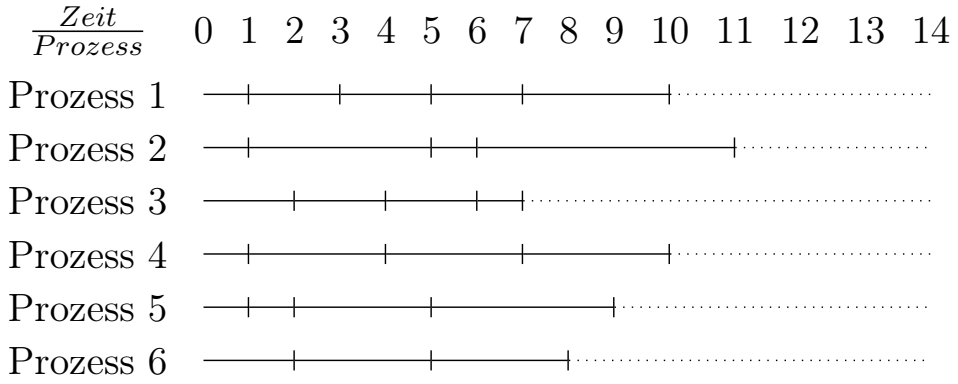


Abbildung 2.8: Das Beispiel aus Abbildung 2.7 bei asynchroner Implementierung

Dies trifft auf diese Arbeit in besonderem Maße zu, da die Simulationszeit zur Auswertung der Individuen zwischen 30 und 105 Sekunden liegt (vgl. Abb. 1.4).

Kapitel 3

Kopplung

Dieses Kapitel beschreibt die Anbindung des SMS-EMOA an die Simulationsumgebung. Zwei verschiedene Varianten des SMS-EMOA optimieren die Problemstellung; zum einen die unveränderte Version wie in Abschnitt 2.2.4 beschrieben und in [14] vorgestellt, zum anderen die asynchrone Variante aus Abschnitt 2.2.5. Die Ergebnisse beider Varianten werden in Kapitel 4 vorgestellt und verglichen. In der Standard-Variante arbeiten mehrere identische SMS-EMOAs getrennt voneinander parallel. Jeder Algorithmus erzeugt seine eigene optimale Lösung.

Um die Ergebnisse vergleichbar zu gestalten, bekommen beide Varianten das gleiche Budget (gleiche Ressourcen). In diesem Fall bedeutet es, dass sie die gleiche Anzahl von Auswertungen zur Verfügung haben, um das Optimum zu erreichen. Allein diese Begrenzung hat einen Abbruch der Optimierung zur Folge. Sie bildet das einzige Abbruchkriterium. Laufen a SMS-EMOA parallel und haben ein Budget von b Auswertungen, ergibt sich für die asynchrone Implementierung ein Budget von $c = a \cdot b$ Auswertungen bei a Threads. a gibt dabei den Grad der Parallelisierung an. Dieser ist durch die maximale Anzahl simulierter Roboter R beschränkt. Theoretisch ist $a > r$ möglich, bringt aber keine bessere Nutzung vorhandener Ressourcen (Simulatoren). Die Populationsgröße μ_a des asynchronen SMS-EMOA wird ebenfalls angepasst. Für eine Populationsgröße μ_s der Standard-Variante ergibt sie sich aus $\mu_a = a \cdot \mu_s$. Dadurch soll eine vergleichbare Anzahl möglicher Lösungen generiert werden.

3.1 Operatoren des SMS-EMOA

Neben dem gleichen Abbruchkriterium (maximale Anzahl Auswertungen) nutzen beide Varianten identische Operatoren zur Selektion der Elter-Individuen und Variation. Bei der Variation wird sowohl eine *Rekombination (crossover)* als auch eine *Mutation (mutation)* durchgeführt.

Elternselektion

Die Elternselektion wählt aus der aktuellen Population zwei Individuen aus. Aus diesen Individuen werden Nachkommen erzeugt. Die Elter-Individuen werden gleichverteilt aus der gesamten Population gewählt. Eine andere Möglichkeit ist die Elter-Individuen nur aus der besten Front zu wählen. Von dieser Variante wurde hier abgesehen, um in der Auswahl bezüglich der Rekombinationsmöglichkeiten nicht zu sehr eingeschränkt zu sein.

Rekombination

Zur Rekombination der beiden selektierten Individuen wird das *Simulated Binary Crossover (SBX)* verwendet. SBX transportiert das Verhalten und die Eigenschaften des *binary single-point crossover* für binäre Suchräume in reelle Suchräume. Es werden zwei Kinder-Individuen aus zwei Eltern erzeugt. Die Position der Kinder im Suchraum ist abhängig vom Abstand der Eltern im Suchraum. Sind die Eltern weit voneinander entfernt, liegen die erzeugten Kinder wiederum weit von ihnen entfernt. Bei kleinem Abstand zwischen den Eltern werden Kinder in der Nähe der Eltern erzeugt. Die Distanz zwischen beiden Elter-Kind-Paaren ist, wie beim binären Operator, identisch. Außerdem liegen beide Kinder entweder innerhalb des von den Eltern eingeschlossenen Bereichs (Kontraktion) oder außerhalb dessen (Expansion) [10]. Der Verteilungsfaktor β_i gibt die Stärke der Kontraktion oder Expansion für die Kinder k_i^1 und k_i^2 und die Eltern e_i^1, $e_i^2 \in \Re$ in jeder Dimension i an:

$$\beta_i = \left| \frac{k_i^1 - k_i^2}{e_i^1 - e_i^2} \right| \tag{3.1}$$

Beim Erzeugen von Nachkommen soll die Wahrscheinlichkeitsverteilung von β_i durch folgende Funktion bestimmt sein:

$$P(\beta_i) = \begin{cases} 0.5(n+1)\beta_i^n \ , & \text{für } \beta_i \leq 1, \\ 0.5(n+1)\frac{1}{\beta_i^{n+2}} \ , & \text{sonst} \end{cases} \tag{3.2}$$

In Formel 3.2 ist n der Distributionsindex. Dieser muss positiv sein ($n \in \Re^+$) und verändert die Wahrscheinlichkeit, mit der die Kinder nahe der Eltern erzeugt werden. Ein hoher Distributionsindex erhöht diese Wahrscheinlichkeit.

SBX erzeugt den Wert des Kindes in Dimension i wie in [10] beschrieben in drei Schritten:

1. Erzeuge eine Zufallszahl z aus $[0, 1]$
2. Finde β_i^* für das die kumulierte Wahrscheinlichkeit gleich z ist

$$\int_0^{\beta_i^*} P(\beta_i) d\beta_i = z$$

3. Berechne mit β_i^* die Werte der Kinder k^1 und k^2 in Dimension i aus

$$k_i^1 = 0.5[(e_i^1 + e_i^2) - \beta_i^* \left|e_i^2 - e_i^1\right|]$$
$$k_i^2 = 0.5[(e_i^1 + e_i^2) + \beta_i^* \left|e_i^2 - e_i^1\right|]$$

Das SBX ist, wie Deb und Beyer in [9] zeigen, *selbstanpassend*, wodurch keine Anpassungsregel benötigt wird. Der Vorteil der Selbstanpassung ist, dass keine zusätzlichen Strategieparameter zum EA hinzukommen.

Mutation

Nach der Rekombination wird eines der beiden Kinder mittels polynomieller Mutation verändert. Es wird nur ein Kind benötigt, da der SMS-EMOA eine $(\mu+1)$-Selektionsstrategie nutzt. Die polynomielle Mutation verwendet, wie SBX, einen Distributionsindex n, um die Veränderung zu beeinflussen. Falls die zu verändernde Variable begrenzt ist, wird die Wahrscheinlichkeitsverteilung so angepasst, dass keine Werte außerhalb der Grenzen erzeugt werden [6].

Der mutierte Wert k der Variablen $e \in [a, b]$ wird mit Hilfe der Zufallszahl $z \in [0, 1]$ wie folgt berechnet:

$$k = \begin{cases} e + \overline{\delta_L}(e - a), & \text{für } z \leq 0.5, \\ e + \overline{\delta_R}(b - e), & \text{sonst} \end{cases}$$

Die Parameter $\overline{\delta_L}$ und $\overline{\delta_R}$ bestimmen die Weite der Mutation in Richtung der unteren Grenze a bzw. der oberen Grenze b. Sie werden durch die folgenden Funktionen berechnet:

$$\overline{\delta_L} = (2z)^{\frac{1}{1+n}} - 1, \qquad \text{für } z \leq 0.5,$$
$$\overline{\delta_R} = 1 - (2(1-z))^{\frac{1}{1+n}}, \quad \text{sonst}$$

Beide Operatoren dienen der Exploration des Suchraums. Bei der Rekombination durch SBX sollen gute Eigenschaften (Parameter) verschiedener Individuen kombiniert werden. Die Mutation soll durch ungerichtete Veränderungen einzelner Entscheidungsvariablen in weitere unerforschte Bereiche vorstoßen.

In den verwendeten Algorithmen gibt es neben den Distributionsindizes, welche für beide Operatoren auf 20 gesetzt wurden, Ausführungswahrscheinlichkeiten der Operatoren. Das bedeutet, dass ein Operator nur mit einer bestimmten Wahrscheinlichkeit auf einer Entscheidungsvariablen ausgeführt wird.

3.2 Problemanalyse

Die Unterteilung des zu Grunde liegenden Problems wurde bereits in Abschnitt 1.1.1 beschrieben. Es soll unter verschiedenen Gesichtspunkten optimiert werden. Denkbare Kriterien sind:

- Fahrzeit oder durchschnittliche Geschwindigkeit
- Distanz zu Hindernissen
- Laufruhe/Glätte bzw. kein „stotterndes“ Fahrverhalten
- Genauigkeit am Zielpunkt
- Abweichung des gefahrenen Weges vom geplanten Pfad

Die Punkte sind nicht miteinander vereinbar und bilden ein mehrkriterielles Optimierungsproblem. Die Genauigkeit am Zielpunkt und die Abweichung des gefahrenen Weges vom geplanten Pfad werden zunächst nicht berücksichtigt. Somit bleiben drei Kriterien nach denen optimiert wird.

3.2.1 Zielfunktionen

Einer der wesentlichen Gesichtspunkte ist die durchschnittliche Geschwindigkeit des Roboters bzw. seine Fahrzeit, denn das Haupteinsatzgebiet des *Adept Lynx* ist der Warentransport. Je schneller Güter transportiert werden, z. B. vom Lager zur Maschine oder umgekehrt, desto höher ist der Durchsatz und desto mehr Fahrten können in einem bestimmten Zeitraum erfolgen. Für den Betrieb ergeben sich zwei Vorteile: Eine erhöhte Produktivität durch die gesteigerte Automatisierung und eine Kostenersparnis durch Senkung der Lohnkosten.

Darüber hinaus ist ein kollisionsfreier Transport unabdingbar. Kollisionen könnten z. B. mit Menschen entstehen, da der Roboter in deren Arbeitsraum eingesetzt werden kann. Um Verletzungen der Personen und Beschädigungen der vom Roboter transportierten Waren zu vermeiden, muss der *Adept Lynx* Hindernisse so weiträumig umfahren, dass eine Kollision ausgeschlossen werden kann. Dazu darf eine gewisse Distanz zu Hindernissen nicht unterschritten werden.

Um die Güter besser zu schützen, soll der *Adept Lynx* eine besonders sanfte Fahrweise haben, d. h. er darf nicht „stotternd“ fahren. Beim Transport von Lithiumwafern in der Halbleiterindustrie können diese leicht beschädigt werden, wenn sie in ihren Boxen bei der Fahrt umher rutschen. Je weicher die Fahrweise ist, desto geringer ist die Gefahr, die transportierte Ware zu beschädigen. Außerdem beansprucht dies die Motoren der Antriebsräder nicht so stark, was sich wiederum positiv auf deren Lebensdauer auswirkt.

Diese drei essentiellen Kriterien spiegeln sich in den folgenden Zielfunktionen wider:

- Fahrzeit

$$f(t) = t_{Ziel} - t_{Start} \tag{3.3}$$

- Laufruhe/Glätte → Summe der Geschwindigkeitsänderungen
  - Translational (Verschiebung; vorwärts/rückwärts)

$$g(v) = \sum_{t=1}^{N-1} (v_{trans_t} - v_{trans_{t-1}})^2 \tag{3.4}$$

  - Rotational (Drehung)

$$h(v) = \sum_{t=1}^{N-1} (v_{rot_t} - v_{rot_{t-1}})^2 \tag{3.5}$$

  mit Geschwindigkeit v und Anzahl N diskreter Zeitpunkte.

- Kleinste Distanz zu Hindernissen

$$dist(x) = \min(X) \tag{3.6}$$

  mit X als Menge aller minimalen Abstände zu Hindernissen:

$$X = \{\min(X_1), \min(X_2), \ldots, \min(X_N)\}$$

  mit X_t als Menge aller Distanzen zu Hindernissen zum Zeitpunkt t.

Wie in Gleichung 3.3 zu sehen ist, wird die Fahrzeit der durchschnittlichen Geschwindigkeit vorgezogen. Zum einen, weil es sich um ein Minimierungsproblem handelt und dieser Wert direkt verwendet werden kann, zum anderen, weil die Fahrzeit sehr einfach um die Planungszeit erweitert werden kann. Die Fahrzeit bezieht sich dementsprechend nicht auf die reine Zeit in welcher der Roboter sich bewegt, sondern beinhaltet zudem die Zeit während der der Roboter noch auf der Stelle steht und den Pfad zum Ziel plant.

Die Gleichungen 3.4 und 3.5 beschreiben die Glätte des Pfades. Das Kriterium besteht aus zwei Zielfunktionen, da je nach Aufbau und Applikation beide Einflüsse evtl. unterschiedlich gewichtet werden. Beide Zielfunktionen beziehen sich auf die Geschwindigkeit des Roboters bzw. vielmehr deren Änderung. Ein ruhiges Fahrverhalten ist somit abhängig von der Beschleunigung. Diese kann sowohl in der Simulation als auch beim realen Roboter einfach gemessen werden.

Gleichung 3.6 beschreibt den minimalen Abstand zu einem Hindernis während der gesamten Fahrt. Dieser Abstand fließt in die Optimierung allerdings nicht als Zielfunktion mit ein. Er gilt vielmehr als Restriktion. Kein Hindernis darf näher zum Roboter sein als

50mm. Sollte dies dennoch eintreten, wird der Parametersatz als unzulässig gekennzeichnet. Das hat in diesem Fall zur Folge, dass die drei übrigen Zielfunktionen (3.3, 3.4, 3.5) auf einen Maximalwert gesetzt werden. Als eine Art Straffunktion wird zu diesem Maximalwert die Schwere der Verletzung der Grenze hinzu addiert. Damit wird der SMS-EMOA aus dem Bereich der unzulässigen Parameter heraus gedrängt, falls er diesen Bereich betritt. Die errechnete Distanz zum nächsten Hindernis kann dabei sogar negativ sein, was gleichbedeutend mit einer Kollision wäre, da das Hindernis in diesem Fall innerhalb des Roboters läge. Es ist möglich, dass die Lösung aus Parametern besteht, die es dem Roboter nicht ermöglichen sich in der Umgebung zu bewegen. In diesem Fall werden die Werte der Zielfunktionen auf den schlechtesten möglichen Wert gesetzt. Dieser besteht aus der Summe des maximalen Straffunktionswertes und dem o. g. Maximalwert.

3.2.2 Parameterbeschreibung

Das zu optimierende Problem besteht, wie zuvor in Abschnitt 1.1.1 beschrieben, aus zwei Phasen; der Pfadplanung und der Pfadverfolgung. Jede der Phasen wird von verschiedenen Parametern beeinflusst. Insgesamt existieren in der aktuellen Softwareversion 80 Parameter zur Pfadplanung. Die Optimierung ist auf die 16 wichtigsten Parameter beschränkt. Andere Variablen, die virtuelle Bereiche der Karte wie z. B. bevorzugte Linien parametrisieren, werden nicht berücksichtigt. Die verwendeten Parameter sind in vier Gruppen unterteilt; *Pfadplanung*, *Pfadverfolgung*, *Roboterdynamik* und *Roboterdynamik bei der Zielanfahrt*. Der Wertebereich, in dem die Parameter optimiert werden dürfen, ist in eckigen Klammern angegeben $[min, max]$.

Pfadplanung

Die Pfadplanung arbeitet, wie in Abschnitt 1.1.1 beschrieben, nach dem Prinzip der Kostenminimierung. Jede Zelle bekommt einen Kostenwert. Der Pfad mit den geringsten Kosten vom Start- zum Zielpunkt wird verwendet. Die Kosten sind dabei von sowohl realen als auch virtuellen Hindernissen, wie verbotenen Zonen, abhängig. Alle Parameter sind in mm angegeben.

PlanRes $[25, 250]$ Die Auflösung des Gitters (Karte) auf dem der Roboter seinen Pfad plant. Die Gitterzellen sind Quadrate, deren Kantenlänge durch diesen Parameter definiert ist. Je größer der Wert, desto ungenauer, aber schneller ist die Planung des Pfades. Eine Halbierung der Kantenlänge resultiert in dem vierfachen Aufwand zur Berechnung des Pfades. Bei zu großen Werten kann es dazu führen, dass enge Bereiche nicht zu passieren sind.

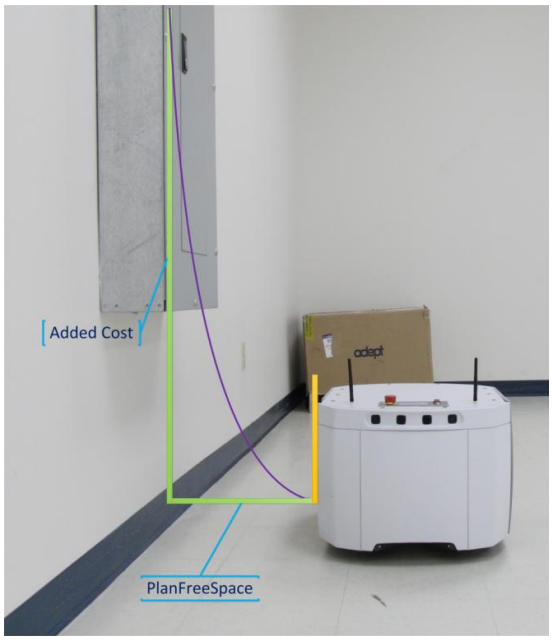


Abbildung 3.1: Auswirkung des Parameter *PlanFreeSpace* auf die Kosten einer Zelle in der Karte. Bei der Pfadplanung werden jeder Zelle Kosten, basierend auf ihrem Abstand zum nächsten Hindernis zugewiesen. Jede Zelle, die weniger als die Hälfte der Roboterbreite von einem Hindernis entfernt liegt, ist für den Roboter durch unendliche Kosten als nicht passierbar abgebildet. Zellen mit halber Roboterbreite Abstand zum nächsten Hindernis gelten als passierbar, kosten aber den maximalen Betrag. Die Kosten aller Zellen, die weiter von einem Hindernis entfernt liegen, werden mit steigender Entfernung verringert. PlanFreeSpace gibt die Distanz an, ab der die Kosten nicht weiter abnehmen.

PlanFreeSpace $[0, 10000]$ Der bevorzugte minimale Abstand seitlich des Roboters zu Hindernissen. Gitterzellen die näher zu Hindernissen sind als dieser Wert, können vom Roboter durchfahren werden, allerdings zu erhöhten Kosten (vgl. Abb. 3.1).

Pfadverfolgung

Bei der Pfadverfolgung wird der Roboter um eine virtuelle Hülle erweitert. Darin gibt es drei Bereiche, wobei in jedem Bereich andere Verzögerungen (Bremswirkungen) gelten. Die Bereiche sind unterteilt in *FrontClearance*, *FrontPadding* und *SideClearance*. Das *FrontPadding* und die *SideClearance* sind geschwindigkeitsabhängig. Sie wachsen und schrumpfen bis zu einem gewissen Grad (vgl. Abb. 3.2); Die Variation der Bereiche kann über die Parameter definiert werden. Die Geschwindigkeiten, zwischen denen die Bereiche verändert werden, sind ebenfalls parametrierbar. Um weitere Komplexität des Problems zu vermeiden, bleiben sie auf den Standardwerten; $SlowSpeed = 300\frac{mm}{s}$, $FastSpeed = 1000\frac{mm}{s}$. Alle Parameter außer *NumOfSplinePoints* sind in mm angegeben. *NumOfSplinePoints* ist als einziger Parameter einheitlos und ganzzahlig.

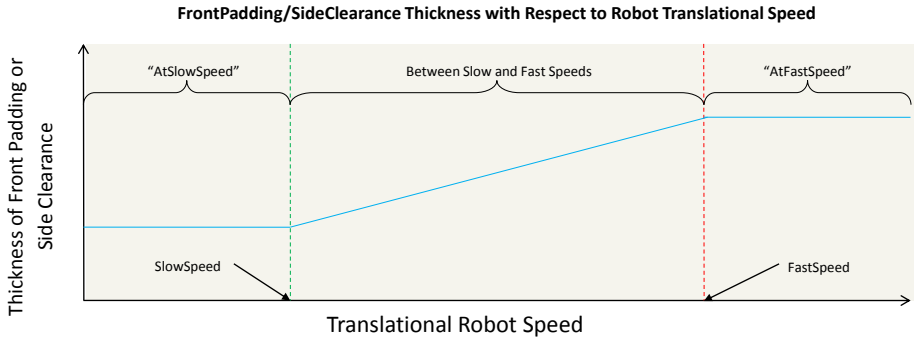


Abbildung 3.2: Einfluss der translatorischen Geschwindigkeit des Roboters auf die Größe des FrontPadding bzw. der SideClearance

FrontClearance $[10, 1000]$ Der Bereich vor dem Roboter in dem eine Notbremsung durchgeführt wird, falls sich ein Hindernis darin befindet.

FrontPaddingAtSlowSpeed $[0, 1000]$ Erweitert den Bereich der FrontClearance um einen weiteren Bereich in dem gebremst wird, falls sich ein Hindernis darin befindet. Dieser Parameter gibt die Größe des Bereichs bei langsamer Geschwindigkeit an.

FrontPaddingAtFastSpeed $[0, 2000]$ Erweitert den Bereich der FrontClearance um einen weiteren Bereich in dem gebremst wird, falls sich ein Hindernis darin befindet. Dieser Parameter gibt die Größe des Bereichs bei hoher Geschwindigkeit an.

SideClearanceAtSlowSpeed $[10, 1000]$ Der Bereich seitlich des *Adept Lynx* in dem eine Notbremsung durchgeführt wird, falls sich ein Hindernis darin befindet. Dieser Parameter gibt die Größe des Bereichs bei langsamer Geschwindigkeit an.

SideClearanceAtFastSpeed $[0, 2000]$ Der Bereich seitlich des *Adept Lynx* in dem eine Notbremsung durchgeführt wird, falls sich ein Hindernis darin befindet. Dieser Parameter gibt die Größe des Bereichs bei hoher Geschwindigkeit an.

NumOfSplinePoints $[3, 5]$ Die Anzahl der Stützpunkte des B-Spline. Der B-Spline wird zur Berechnung des lokalen Pfades verwendet. Er sorgt dafür, dass der *Adept Lynx* sanft um Kurven und Hindernisse fährt.

Abbildung 3.3 stellt die einzelnen Bereiche dar. Der Padding-Bereich vor der *FrontClearance* erlaubt eine sanftere Verzögerung, wenn ein Hindernis entdeckt wird. Dadurch bremst der Roboter erst dann stark ab, wenn die akute Gefahr besteht mit dem Hindernis zu kollidieren. Zuvor versucht er das Hindernis mit gedrosselter Geschwindigkeit zu passieren. Das resultiert in einer ruhigeren Fahrweise.

Abbildung 3.4 zeigt, dass größere Werte nicht immer Vorteile bringen. Bei zu großer Differenz der Parameter wirkt sich dies kontraproduktiv auf das Fahrverhalten aus.

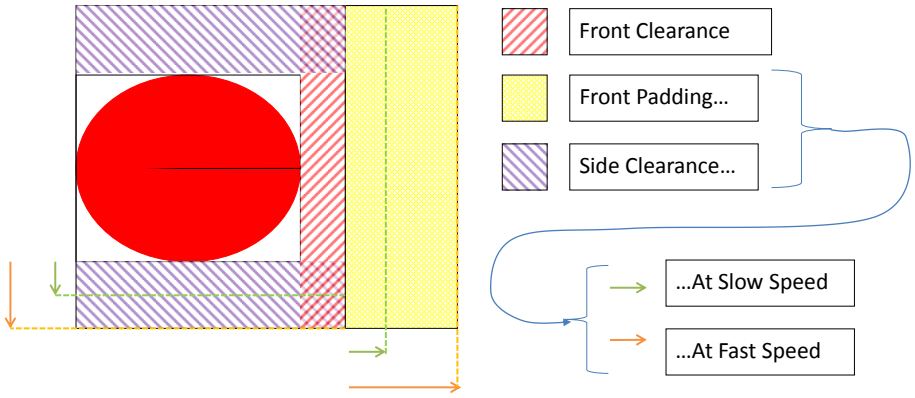


Abbildung 3.3: Einfluss der Pfadverfolgungsparameter auf die Hülle des Roboters.

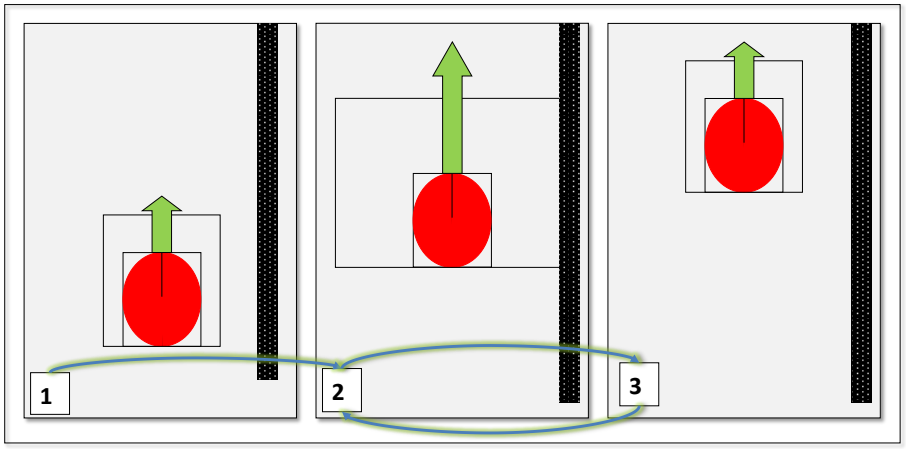


Abbildung 3.4: Veranschaulichung eines möglicherweise stotternden Fahrverhaltens des *Adept Lynx* bei zu großer Differenz zwischen *SideClearanceAtSlowSpeed* und *SideClearanceAtFastSpeed*. Während der Roboter beschleunigt, vergrößert sich die SideClearance bis sie ein Hindernis (in diesem Fall die Wand) enthält. Das erkannte Hindernis sorgt dafür, dass der Roboter bremst, wodurch die SideClearance automatisch schrumpft. Als Resultat daraus darf der Roboter erneut beschleunigen. Dieser Zyklus wird wiederholt durchlaufen und der Roboter bekommt ein „stotterndes“ Fahrverhalten.

Roboterdynamik

Diese Parameter beziehen sich auf die Dynamik des Roboters. Es sind Geschwindigkeiten, Beschleunigungen und Verzögerungen, die bei der Pfadverfolgung genutzt werden.

MaxSpeed $[300, 1900]$ Die maximale nach vorne gerichtete Geschwindigkeit (in $\frac{mm}{s}$), mit der der Roboter dem Pfad folgt.

MaxRotSpeed $[90, 180]$ Die maximale Rotationsgeschwindigkeit (in $\frac{deg}{s}$), die bei der Pfadverfolgung erreicht wird.

DrivingTransAccel $[300, 1500]$ Die Beschleunigung (in $\frac{mm}{s^2}$) des Roboters.

DrivingTransDecel $[300, 1500]$ Die negative Beschleunigung (in $\frac{mm}{s^2}$), mit der der Roboter bremst.

DrivingRotAccel $[100, 360]$ Die Rotationsbeschleunigung (in $\frac{deg}{s^2}$) des Roboters.

DrivingRotDecel $[100, 360]$ Die negative Rotationsbeschleunigung (in $\frac{deg}{s^2}$), mit der der Roboter bremst.

Die Parameter *DrivingTransAccel* und *DrivingTransDecel* sowie *DrivingRotAccel* und *DrivingRotDecel* werden bei der Optimierung zu jeweils einem Parameter zusammengefasst. Das bedeutet, dass die Verzögerung genauso stark ausfällt wie die Beschleunigung.

Roboterdynamik bei der Zielanfahrt

Zur Zielanfahrt wechselt der Roboter in einen anderen Modus als bei der Pfadverfolgung. Dieser Modus, der dynamisch aus der normalen Pfadverfolgung heraus aktiviert wird, nutzt die unten aufgeführten Parameter. Dabei spielt die Differenz aus der aktuellen und hier gewählten Geschwindigkeit eine Rolle. Die Dynamik des Roboters ist für diesen Modus gesondert parametrierbar. Um ein sanfteres und genaueres Erreichen der Zielposition zu ermöglichen, kann es von Vorteil sein, die Geschwindigkeiten und die Beschleunigungen bei der Zielanfahrt zu reduzieren.

GoalSpeed $[100, 1900]$ Die maximale Geschwindigkeit (in $\frac{mm}{s}$) des Roboters bei der Zielanfahrt.

GoalRotSpeed $[50, 180]$ Die maximale Rotationsgeschwindigkeit (in $\frac{deg}{s}$) des Roboters bei der Zielanfahrt.

GoalTransAccel $[100, 1000]$ Die Beschleunigung des Roboters (in $\frac{mm}{s^2}$) bei der Zielanfahrt.

GoalTransDecel $[100, 1000]$ Die negative Beschleunigung (in $\frac{mm}{s^2}$), mit der der Roboter bei der Zielanfahrt bremst.

GoalRotAccel $[100, 360]$ Die Rotationsbeschleunigung (in $\frac{deg}{s^2}$) des Roboters bei der Zielanfahrt.

GoalRotDecel $[100, 360]$ Die negative Rotationsbeschleunigung (in $\frac{deg}{s^2}$), mit der der Roboter bei der Zielanfahrt bremst.

Die Parameter *GoalTransAccel* und *GoalTransDecel* sowie *GoalRotAccel* und *GoalRotDecel* werden bei der Optimierung zu jeweils einem Parameter zusammengefasst. Das bedeutet, dass die Verzögerung genauso stark ausfällt wie die Beschleunigung.

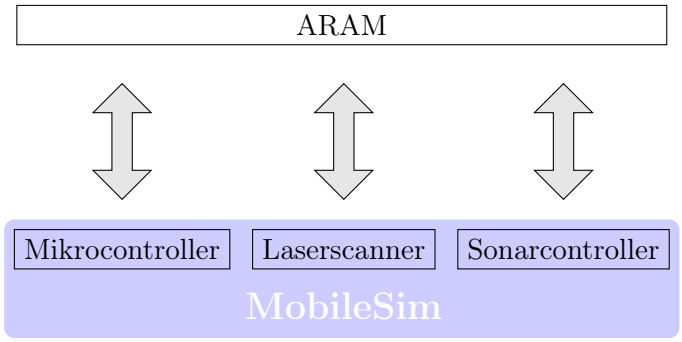


Abbildung 3.5: Softwarearchitektur des *Adept Lynx*.

3.3 Simulator

Im *Adept Lynx* arbeiten im Wesentlichen zwei Komponenten zusammen; die navigierende Software und ein Mikrocontroller. Das Hauptprogramm, *ARAM*, trifft Entscheidungen welche Aufgabe der Roboter ausführt und navigiert zum nächsten Zielpunkt. Der Mikrocontroller setzt die Befehle von ARAM um und steuert den Roboter. Diese Befehle bilden die Eingabe in Form von anzusteuernden Geschwindigkeiten. Der Ursprung der Kommandos kann innerhalb ARAM variieren. Es können einfache, vordefinierte Bewegungen wie eine geradlinige Fahrt sein, aber auch Geschwindigkeiten, die aus dem autonomen Navigationssystem generiert werden, um dem dynamisch geplanten Pfad zu folgen. Als Ausgabe liefert der Mikrocontroller eine Positionsschätzung auf Basis der Rad-Encoder sowie des Gyroskops, die sogenannte Odometrie.

Die Simulationssoftware, *MobileSim*, ersetzt den Mikrocontroller des realen Roboters. Die Kommunikation zwischen ARAM und MobileSim ist identisch zur Kommunikation mit dem Mikrocontroller. Der einzige Unterschied ist, dass die Verbindung zu MobileSim über TCP läuft während die Verbindung zum Mikrocontroller eine serielle Schnittstelle nutzt. MobileSim ersetzt neben der Steuerungsschicht auch den Laserscanner und Sonarcontroller (vgl. Abb. 3.5). Der Simulator beinhaltet sowohl ARAM als auch MobileSim. Für andere Programme ist er damit nicht direkt von einem echten Roboter zu unterscheiden. Die Kommunikation mit dem realen Roboter ist ohnehin in den meisten Fällen eine Kommunikation mit ARAM.

3.3.1 MobileSim Arbeitsweise

MobileSim basiert auf der *Stage Bibliothek* [25]. Es wandelt die Karte auf der der Roboter arbeitet, in eine Stage-Umgebung um und platziert den Roboter bzw. das Model des Roboters in der Umgebung. Die Karte der Stage-Simulation besteht aus zweidimensionalen Punkten und Linien. Diese sind von allen simulierten Sensoren wahrnehmbar und der Roboter kann mit ihnen kollidieren. In dieser Arbeit wird ein Roboter mit einem Laserscanner

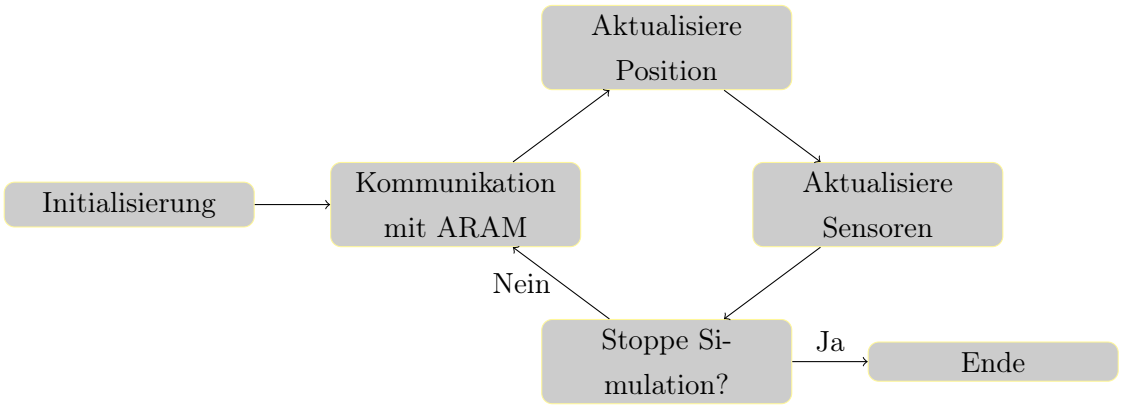


Abbildung 3.6: Schematische Darstellung der Simulation

simuliert, welcher im Model des *Adept Lynx* verankert ist und sich somit relativ zu diesem mit in der simulierten Umgebung bewegt.

Nach der Initialisierung, dem Laden der Karte und der Modelle, beginnt eine zeitkontinuierliche Simulation. Die Frequenz mit der MobileSim den Zustandsraum bestehend aus der Umgebung, dem Robotermodell und den Laserwerten aktualisiert, beträgt 10Hz. Die Kommunikation mit ARAM ist von dieser Frequenz unbeeinflusst; ihre Frequenz zum Datenaustausch ist jedoch mit ihr identisch. Der schematische Ablauf der Simulation ist in Abbildung 3.6 dargestellt. Wie zu sehen ist, empfängt MobileSim zunächst Geschwindigkeitskommandos von ARAM und berechnet daraus die neue Position des Roboters, ähnlich zur Positionsschätzung des Mikrocontrollers. Anschließend beginnt das Sensorupdate auf Basis der neuen Position. Sofern die Simulation nicht gestoppt wird, sendet MobileSim die neu berechnete Position, inklusive einer möglichen Kollision, sowie die Werte des Laserscanners an ARAM und empfängt im Gegenzug die Geschwindigkeiten für die nächste Aktualisierung.

Positionsupdate

Durch seinen Differentialantrieb besitzt der *Adept Lynx* zwei Freiheitsgrade. Für ein Positionsupdate benötigt er daher zwei Geschwindigkeiten - translatorisch und rotatorisch. Diese Soll-Geschwindigkeiten werden gegebenenfalls auf die in der Initialisierung gesetzten Maximalgeschwindigkeiten heruntergesetzt. Im nächsten Schritt berechnet das Verfahren die nötige Beschleunigung, welche ebenfalls auf ihren maximalen Wert begrenzt wird. Falls nötig wird die aktuelle Geschwindigkeit um die zulässige, benötigte Beschleunigung er-

höht bzw. vermindert. Mit Hilfe dieser neu gesetzten Geschwindigkeit wird eine mögliche Kollision detektiert und anschließend die neue Position wie folgt berechnet:

$$
\begin{aligned}
m_x &= (v_x \cdot dt) \cdot (1 + error_x) \\
m_y &= (v_y \cdot dt) \cdot (1 + error_y) \\
m_\theta &= (v_\theta \cdot dt) \cdot (1 + error_\theta) \\
pose^x_{t+1} &= pose^x_t - m_y \cdot \cos(\theta) - m_x \cdot \sin(\theta) \\
pose^y_{t+1} &= pose^x_t + m_x \cdot \cos(\theta) + m_y \cdot \sin(\theta) \\
pose^\theta_{t+1} &= pose^x_t + m_\theta
\end{aligned}
$$

Hierbei gibt v_* die Geschwindigkeit in x oder y Richtung oder rotatorisch v_θ an. Es ist zu beachten, dass v_y immer 0 ist, da der *Adept Lynx* nicht seitwärts fahren kann. Die verstrichene Zeit seit der letzten Aktualisierung ist durch dt gegeben. Die Orientierung des Roboters auf der Karte ist mit θ angegeben. Durch $error_x$, $error_y$ und $error_\theta$ können Fehler in der Odometrie simuliert werden. Es handelt sich dabei um einen systematischen, statischen Fehler. In der Stage-Simulation wurde diese Form gewählt, da Stage auf die Simulation von Roboterflotten ausgelegt ist und die Berechnung des Positionsupdates einzelner Roboter nicht aufwendig sein darf.

Sensorupdate

Das Modell des Laserscanners ist ein einfaches Sensormodell zur Distanzberechnung. Die Distanzen werden in Intervallen über das Sichtfeld des Lasers berechnet. Diese Intervalle ergeben sich aus der Anzahl der Samples und dem Öffnungswinkel, welche während der Initialisierung gesetzt werden. Beim simulierten Laserscanner beträgt das Sichtfeld 270°. Bei 540 Samples ergeben sich Intervalle von 0.5°, entsprechend den Werten des realen Laserscanners. Jedem Sample wird ein Intervall zugeordnet, für das es die Distanz zum nächsten Hindernis berechnet. Zu diesem Winkel wird ein randomisierter, statischer Fehler addiert, bevor, entlang des Strahls vom Laserzentrum aus, nach einem Schnittpunkt mit der Karte oder einem anderen Objekt gesucht wird. Aus diesem Schnittpunkt und dem Zentrum des Laserscanners wird die Distanz berechnet, zu der ein randomisierter, statischer Fehler addiert wird. Falls kein Schnittpunkt innerhalb der maximalen Reichweite gefunden wurde, bekommt dieses Sample diesen Wert zugewiesen.

3.3.2 Simulationsserver

Der Simulationsserver ist ein System, das mehrere Simulatoren bereitstellt. Er besitzt ein Debian Betriebssystem mit *kvm* Paket, das es erlaubt *Virtuelle Maschinen (VMs)* einzusetzen. Dank der hohen Rechenleistung des Systems können bis zu acht Roboter parallel

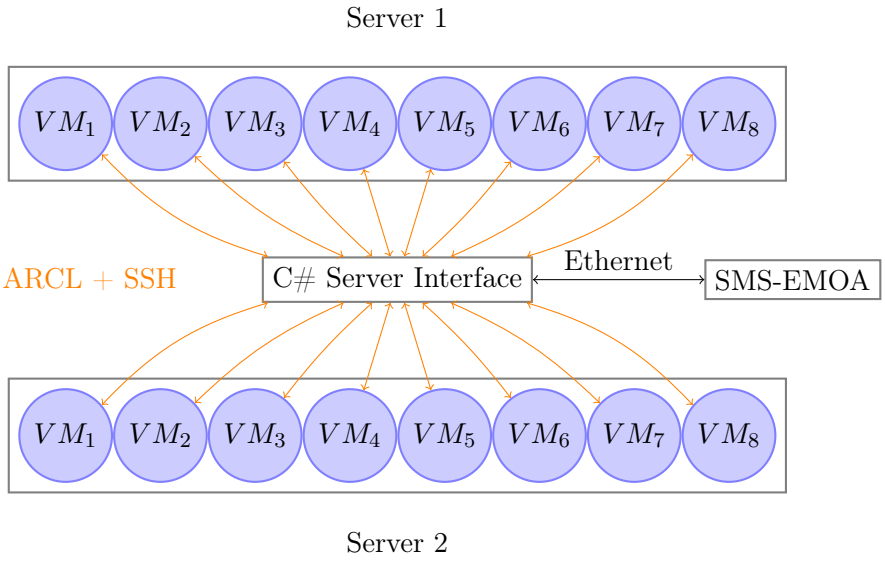


Abbildung 3.7: Zusammenspiel der einzelnen Komponenten

simuliert werden. Jeder Roboter wird in einer VM simuliert. Für diese Arbeit stehen zwei Simulationsserver bereit, wodurch auf 16 simulierten Robotern parallel optimiert werden kann.

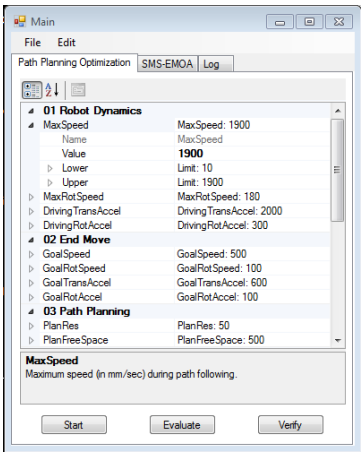
3.4 Infrastruktur

Abbildung 3.7 zeigt die zur Optimierung verwendeten Komponenten. Zur Optimierung wird der von Wessing implementierte SMS-EMOA aus der *jMetal* Bibliothek [13] verwendet. Die asynchrone Variante des SMS-EMOA wurde im Zuge dieser Arbeit ebenfalls innerhalb der jMetal Umgebung implementiert. Zur Auswertung wird die Bibliothek um eine Implementierung der jMeatal-Problem-Klasse erweitert. Diese Klasse wertet die mögliche Lösung aus, indem sie das Individuum an das *C#-Server-Interface* sendet, welches die Auswertung auf einem simulierten Roboter startet. Anschließend sendet es das bewertete Individuum an den SMS-EMOA bzw. die Problem-Klasse zurück.

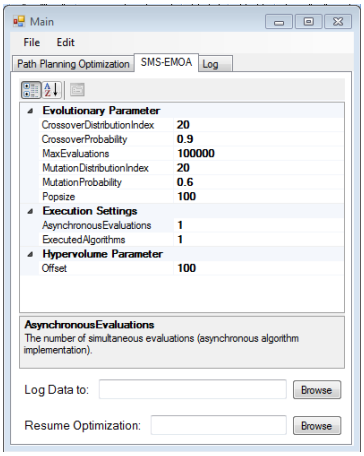
3.4.1 C#-Server-Interface

Das *C#-Server-Interface* ist das Bindeglied zwischen dem simulierten Roboter und dem SMS-EMOA. Zudem stellt es die *Graphical User Interface (GUI)* bereit, um Strategieparameter des SMS-EMOA zu spezifizieren, Parameter- und Restriktions-Grenzen festzulegen und den Optimiervorgang zu starten, stoppen oder fortzusetzen (vgl. Abb. 3.8).

Weiterhin werden Verbindungen zu den einzelnen Robotern spezifiziert (vgl. Abb. 3.9). Zur Evaluation benötigt das C#-Server-Interface zwei Verbindungen zum Roboter; zum

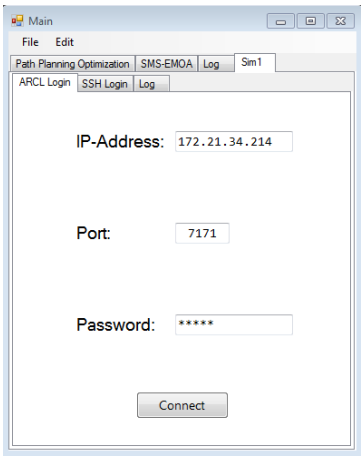


(a) Grafische Oberfläche zum Setzen der Parameter- und Restriktions-Grenzen.

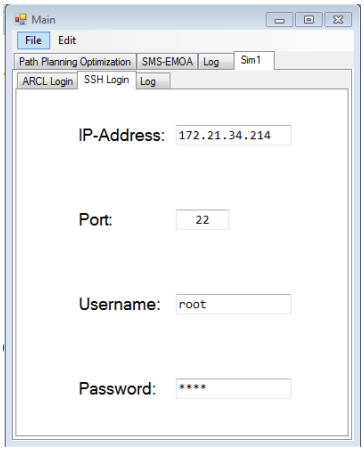


(b) Grafische Oberfläche zur Eingabe von Strategieparametern des SMS-EMOA

Abbildung 3.8: Grafische Oberfläche zum Setzen von Problemspezifikationen (3.8a) und Strategieparametern (3.8b).



(a) Grafische Oberfläche zur Spezifikation der ARCL-Verbindung.



(b) Grafische Oberfläche zur Spezifikation der SSH-Verbindung.

Abbildung 3.9: Grafische Oberfläche zur Einstellung von Verbindungseigenschaften.

einen per *SSH* zum Herunterladen von Dateien, zum anderen über *Advanced Robotics Command Language (ARCL)*, um den Roboter zu steuern.

In seiner Aufgabe als Bindeglied verwaltet das Interface die verfügbaren Roboter und wertet vom SMS-EMOA empfangene Individuen auf ihnen aus. Zur Kommunikation mit dem Optimieralgorithmus bietet es einen *TCP-Server*. Mehrere Algorithmen können sich an diesem Server mit ihrer ID anmelden, um ihre Individuen bewerten zu lassen. Zur Kommunikation gibt es vorgefertigte Schlüsselwörter, die angeben, welche Art von Daten übertragen werden und was mit ihnen geschehen soll. Der Aufbau der Nachrichten ist in Anhang A.1 angegeben. Die vom Nutzer gesetzten Strategieparameter sowie die Parameter

und ihre Grenzen, werden dem Algorithmus, nach seiner Anmeldung an den Server, ebenso mitgeteilt, wie die Zielfunktionen bzw. die Dimension des Lösungsraums.

Evaluation

Zur Evaluation eines Individuums transferiert der SMS-EMOA eine Evaluations-Nachricht mit dem Individuum an das C#-Server-Interface. Dieses hängt das Individuum zunächst in eine *synchronisierte FIFO-Queue*. Die Queue wird von einem Organisationsprozess gelesen und abgearbeitet. Sobald ein Roboter verfügbar ist, d. h. kein anderes Individuum auswertet, wird ein Individuum aus der Queue entnommen und auf dem Roboter ausgewertet (vgl. Alg. 3.2). Algorithmus 3.1 beschreibt den Auswertungsprozess.

Algorithmus 3.1 Evaluation eines Individuums

Eingabe: individual, robot, finishedQueue

```
1:  if not isConnected(robot) then
2:      connect(robot)                    // Verbinde mit Roboter, falls nicht verbunden
3:  end if
4:  setParameter(individual)              // Setze Parameter zur Auswertung
5:  moveRobot(startPosition)              // Fahre Roboter auf die Startposition
6:  startNewLog(robot)                    // Starte neue Datenaufnahme
7:  startPatrol(robot)                    // Beginne Abfahren der Teststrecke
8:  success ← waitToFinish(robot)         // Warte bis der Roboter die Fahrt beendet hat
9:  if success then
10:     individual ← calculateFitness(robot)   // Berechne Zielfunktionswerte
11: else
12:     individual ← worstFitness         // Setze Zielfunktionswerte auf unzulässige Werte;
        schlechtest mögliche Bewertung
13: end if
14: push(finishedQueue, individual)
```

Falls noch keine ARCL-Verbindung zum Roboter besteht, wird zu Beginn eine neue Verbindung aufgebaut. ARCL ist eine Art Programmierschnittstelle für mobile Roboter der Firma Adept. Es ist eine Telnet-Verbindung, über die Kommandos an den Roboter gesendet werden können. Gleichzeitig ist es möglich über ARCL Daten des Roboters, wie z. B. den aktuellen Status, auszulesen. Zu jedem abgesendeten Kommando gibt es eine Antwort.

Nachdem sichergestellt ist, dass eine ARCL-Verbindung besteht, werden darüber die Parameter auf dem Roboter angepasst. Der Roboter wird anschließend auf die Startposition gefahren. Ein simulierter Roboter muss den Weg nicht fahren, sondern wird direkt auf diese Position gesetzt. Danach startet er ein neues Log, mit dessen Hilfe später die

Zielfunktionswerte berechnet werden. In der Datei steht neben einem *monoton steigenden* Zeitstempel der aktuell minimale Abstand zu einem Hindernis sowie die aktuelle translatorische und rotatorische Geschwindigkeit. Diese Daten logt der Roboter alle 100ms. Im Anschluss startet der Roboter die Fahrt über die Teststrecke. Bevor der Roboter vom Startpunkt zum ersten Ziel fährt, sendet er eine Nachricht mit der aktuellen Uhrzeit an das C#-Server-Interface. Nachdem er am letzten Zielpunkt angekommen ist, sendet er erneut eine Nachricht mit der aktuellen Uhrzeit. War die Fahrt erfolgreich, werden die Zielfunktionswerte berechnet.

Die Zielfunktion *Fahrzeit* (3.3) ergibt sich aus den beiden gesendeten Nachrichten. Zur Berechnung der anderen beiden Zielfunktionen benötigt das C#-Server-Interface die Logdatei vom Roboter. Dazu wird die Datei per SSH/SFTP vom Roboter heruntergeladen. Aus diesen Werten werden die beiden übrigen Zielfunktionen berechnet. Sollte zu einem beliebigen Zeitpunkt der minimale Abstand zu einem Hindernis unter die angegebene Grenze fallen, werden alle Zielfunktionswerte auf einen Strafwert gesetzt zu dem die Schwere der Verletzung der Restriktion addiert wird.

Im Fall einer nicht erfolgreichen Fahrt werden alle Zielfunktionen auf den schlechtesten möglichen Wert gesetzt. Dieser ergibt sich aus der Summe des Strafwerts und der maximalen Verletzung der Restriktion. Zuletzt wird in Zeile 14 das bewertete Individuum in eine synchronisierte FIFO-Queue für bewertete Individuen gehängt.

Diese Queue wird von dem Organisationsprozess abgearbeitet. Er entnimmt ein Individuum nach dem Anderen und sendet sie an deren SMS-EMOA zurück (vgl. Alg. 3.2). Dieser kann daraufhin seine Population aktualisieren, ein neues Individuum erzeugen und dieses zur Evaluation per TCP-Verbindung an das C#-Server-Interface senden.

Algorithmus 3.2 Organisation der Evaluation

Eingabe: robotList, toEvaluateQueue, finishedQueue

```
1:  while not stop do
2:    if not isEmpty(toEvaluateQueue) then        // Individuen bereit zur Evaluation?
3:      for all robot in robotList do
4:        if isFree(robot) then                   // Suche nach verfügbarem Roboter
5:          startEvaluation(pop(toEvaluateQueue), robot)   // Starte Evaluation
6:          break                                 // Beende Suche nach freiem Roboter
7:        end if
8:      end for
9:    end if
10:   while not isEmpty(finishedQueue) do         // Wurden Individuen bewertet?
11:     send(pop(finishedQueue))                  // Sende Individuum an Algorithmus zurück
12:   end while
13: end while
```

Kapitel 4

Experimentelle Analyse

Dieses Kapitel besteht aus zwei Hauptteilen. Im ersten Teil wird die Analyse der asynchronen Variante des SMS-EMOA auf vier verschiedenen Testproblemen vorgestellt. Dabei wird die asynchrone Variante zum einen mit dem Standardalgorithmus und zum anderen mit mehreren parallel laufenden SMS-EMOAs verglichen. Im zweiten Teil werden die Simulationsergebnisse der asynchronen Variante und der simultan ausgeführten SMS-EMOAs vergleichend gegenüber gestellt. Die im Simulator optimierten Parameter werden auf dem realen Testparcours überprüft. Dieser Abschnitt analysiert, inwieweit sich die Ergebnisse der Simulation auf den realen Roboter übertragen lassen.

4.1 Analyse des asynchronen SMS-EMOA

Die Experimente zur Untersuchung des asynchronen SMS-EMOA finden auf Testproblemen unterschiedlicher Komplexität statt. Hierbei soll verglichen werden, wie schnell sich die unterschiedlichen Algorithmen der tatsächlichen Pareto-Front annähern und diese abdecken. Die Experimente werden mit Hilfe der jMetal Bibliothek [13] durchgeführt und deren Ergebnisse über mehrere Durchläufe gemittelt.

4.1.1 Testprobleme

Zur Analyse werden zwei zweikriterielle und zwei dreikriterielle Testprobleme verwendet. Als zweikriterielle Testprobleme fungieren die ZDT-Funktionen *ZDT1* und *ZDT3* von Zitzler et al. aus [28]. Als Benchmark auf dreikriteriellen Problemen kommen *DTLZ1* und *DTLZ7* aus den DTLZ-Funktionen von Deb et al. [12] zum Einsatz. Damit wird sowohl im zweikriteriellen als auch im dreikriteriellen jeweils zu einer zusammenhängenden und einer unzusammenhängenden Pareto-Front hin optimiert. Die Pareto-Fronten von ZDT3 und DTLZ7 bestehen aus unzusammenhängenden Teilen.

ZDT-Testprobleme

Die Familie der ZDT-Funktionen besteht aus sechs Problemen, von denen zwei verwendet werden. Alle ZDT-Funktionen sind bikriteriell und bestehen aus drei Funktionen f_1, g, h, die anhand der allgemeinen Form aus [7] konstruiert werden:

$$\begin{aligned} &\text{Minimiere} && T(\boldsymbol{x}) = (f_1(x_1), f_2(\boldsymbol{x})) \\ &\text{mit} && f_2(\boldsymbol{x}) = g(x_2, \ldots, x_m) \cdot h(f_1(x_1), g(x_2, \ldots, x_m)) \\ &\text{und} && \boldsymbol{x} = (x_1, \ldots, x_m) \end{aligned}$$

Der Vektor $\boldsymbol{x}$ enthält die m Entscheidungsvariablen des Problems. Das Ziel ist es, f_1 und f_2 zu minimieren. Die Funktion f_1 hängt ausschließlich von der ersten Entscheidungsvariablen ab. Die restlichen $m-1$ Entscheidungsvariablen werden von der Funktion g genutzt. Die Funktionswerte von f_1 und g sind die Parameter der Funktion h. Die daraus resultierenden Testprobleme unterscheiden sich in den Funktionen f_1, g, h, der Anzahl von Entscheidungsvariablen m sowie deren Wertebereiche.

Die hier verwendeten Probleme sehen wie folgt aus:

Testproblem	**Funktionen**	m	**Eigenschaften**
ZDT1	$f_1(x_1) = x_1$ $g(x_2, \ldots, x_m) = 1 + \frac{9}{m-1} \sum_{i=2}^{m} x_i$ $h(f_1, g) = 1 - \sqrt{f_1/g}$	30	Konvexe Pareto-Front $x_i \in [0, 1]$
ZDT3	$f_1(x_1) = x_1$ $g(x_2, \ldots, x_m) = 1 + \frac{9}{m-1} \sum_{i=2}^{m} x_i$ $h(f_1, g) = 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1)$	30	Konvexe Pareto-Front aus fünf Teilen $x_i \in [0, 1]$

Tabelle 4.1: Die Funktionen ZDT1 und ZDT3. Die Funktionen f_1 und f_2 sind zu minimieren. Es gilt $f_2(\boldsymbol{x}) = g(x_2, \ldots, x_m) \cdot h(f_1, g)$.

DTLZ-Testprobleme

Die DTLZ-Funktionen sind nach ihren Entwicklern Deb, Thiele, Laumanns und Zitzler benannt. Sie umfassen neun Probleme. Die Anzahl der Zielfunktionen kann für jedes Problem der DTLZ-Familie nach Bedarf gewählt werden. In dieser Arbeit werden dreikriterielle DTLZ-Funktionen verwendet.

DTLZ1 Das dreikriterielle DTLZ1-Problem besitzt die Form:

$$\begin{aligned}
&\text{Minimiere} \quad f_1(\boldsymbol{x}) = \frac{1}{2}(1 + g(\boldsymbol{x}_3)) \cdot x_1 \cdot x_2 \\
&\text{Minimiere} \quad f_2(\boldsymbol{x}) = \frac{1}{2}(1 + g(\boldsymbol{x}_3)) \cdot x_1 \cdot (1 - x_2) \\
&\text{Minimiere} \quad f_3(\boldsymbol{x}) = \frac{1}{2}(1 + g(\boldsymbol{x}_3)) \cdot (1 - x_1) \\
&\text{mit} \qquad\quad g(\boldsymbol{x}_3) = 100(|\boldsymbol{x}_3| + \sum_{i=3}^{m} (x_i - 0.5)^2 - \cos(20 \cdot \pi(x_i - 0.5)))
\end{aligned}$$

Der Vektor $\boldsymbol{x} = (x_1, x_2, \boldsymbol{x}_3)$ enthält die m Entscheidungsvariablen des Problems. Die Funktionen f_1, f_2, f_3 sind zu minimieren. Sie hängen von den ersten $M-1$ Variablen des Vektors $\boldsymbol{x}$ ab, wobei M die Anzahl der Zielfunktionen ist; in diesem Fall $M = 3$. Die restlichen $m - M$ Entscheidungsvariablen, enthalten in $\boldsymbol{x}_M$ bzw. $\boldsymbol{x}_3$, nutzt die Funktion $g(\boldsymbol{x}_3)$. Das später verwendete Problem arbeitet mit sieben Entscheidungsvariablen ($m = 7$).

DTLZ7 Das dreikriterielle DTLZ7-Problem besitzt die Form:

$$\begin{aligned}
&\text{Minimiere} \quad f_1(x_1) = x_1 \\
&\text{Minimiere} \quad f_2(x_2) = x_2 \\
&\text{Minimiere} \quad f_3(\boldsymbol{x}) = (1 + g(\boldsymbol{x}_3)) \cdot h(f_1, f_2, g) \\
&\text{mit} \qquad\quad g(\boldsymbol{x}_3) = 1 + \frac{9}{|\boldsymbol{x}_3|} \sum_{i=3}^{m} x_i \\
&\text{und} \qquad\quad h(f_1, f_2, g) = 3 - \sum_{i=1}^{2} \left[\frac{f_i}{1+g} (1 + \sin(3\pi \cdot f_i)) \right]
\end{aligned}$$

Wie bei DTLZ1 sind die Funktionen f_1, f_2 und f_3 zu minimieren. Dabei werden die Funktionen f_1 und f_2 von nur einer der m Entscheidungsvariablen beeinflusst. Die übrigen $m-2$ Variablen verändern die Funktion $g(\boldsymbol{x}_3)$. In den folgenden Tests besitzt das Problem 22 Entscheidungsvariablen.

4.1.2 Vergleich mit paralleler Ausführung

Im ersten Teil der Analyse des asynchronen SMS-EMOA tritt dieser gegen mehrere parallel laufende Standardvarianten an. Alle SMS-EMOA verwenden die gleichen Operatoren zur Selektion, Rekombination und Mutation. Darüber hinaus ist der Offset zur Konstruktion des Referenzpunktes identisch; dieser beträgt 100. Der Grad der Parallelisierung ist für die folgenden Experimente auf fünf festgesetzt. Das bedeutet, dass beim asynchronen SMS-EMOA fünf Prozesse/Threads gleichzeitig auf der gemeinsamen Population arbeiten. Für die parallele Ausführung bedeutet dies, dass fünf SMS-EMOA losgelöst voneinander

das gleiche Problem auf ihrer eigenen Population optimieren. Beide Varianten bekommen das gleiche Budget von 10000 Auswertungen, welche unter den parallel laufenden SMS-EMOA gleichmäßig aufgeteilt werden. Jeder parallel ausgeführte SMS-EMOA hat damit 2000 Auswertungen zur Verfügung. Die Populationsgröße beträgt beim asynchronen SMS-EMOA 40, wobei jede Population der parallelen SMS-EMOA acht Individuen enthält. Damit haben beide Varianten am Ende der Optimierung die Möglichkeit, die gleiche Anzahl an Lösungen zu liefern. Die Lösungen der einzelnen parallel laufenden SMS-EMOAs werden, bevor sie mit den Lösungen der asynchronen Variante verglichen werden, vorgefiltert. Dabei werden alle Lösungen zu einer Menge zusammengefasst. Die nicht-dominierten Lösungen dieser vereinten Menge sind die Lösungen der parallelen SMS-EMOAs. Sollte die Population der asynchronen Variante mit Ende der Optimierung dominierte Lösungen enthalten, werden diese ebenfalls ignoriert.

Sechs Indikatoren bewerten die jeweilige Lösungsmenge der einzelnen Tests. Die Werte der Indikatoren ergeben sich aus zehn unabhängigen Läufen jeder Variante auf dem jeweiligen Testproblem. Es werden die folgenden Indikatoren genutzt:

Hypervolumen Berechnet das Hypervolumen der Lösungsmenge zu einem Referenzpunkt. Der verwendete Referenzpunkt wird problemspezifisch gewählt.

Epsilon Gibt den kleinsten Betrag an, um den die Lösungsmenge verschoben werden muss, damit alle Punkte der Referenzmenge von ihr dominiert werden.

Spread Bewertet die Verteilung der Lösungen im Raum, was der Abdeckung der Pareto-Front entspricht. Die Metrik ist definiert als [11]:

$$\Delta = \frac{d_f + d_l \sum_{i=1}^{n-1} |d_i - \overline{d}|}{d_f + d_l + (n-1) \cdot \overline{d}}$$

Generational Distance (GD) Gibt an, wie weit die Punkte der Lösungsmenge von den jeweils nächstgelegenen Punkten der Referenzmenge entfernt sind.

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

mit n als Anzahl von Lösungen und d_i als Distanz zur nächstgelegenen Lösung der Referenzmenge.

Inverted Generational Distance (IGD) Invertierte Form von GD. Die Metrik berechnet, wie weit jeder Punkt der Referenzmenge von dem ihm nächstgelegenen Punkt der Lösungsmenge entfernt ist.

$$IGD = \sqrt{\frac{\sum_{i=1}^{m} d_i^2}{m}}$$

mit m als Anzahl von Punkten der Referenzmenge und d_i als Distanz zur nächstgelegenen Lösung.

Dauer Die Ausführungsdauer des Algorithmus angegeben in Sekunden.

Dabei gilt: Je größer das Hypervolumen und je kleiner die anderen Indikatoren, desto besser ist der Algorithmus.

Diese Analyse soll Aufschluss über die Leistungsfähigkeit der Algorithmen im Bezug auf die Dauer der Optimierung und ihre Qualität bei einer möglichen Parallelisierung der Auswertungen bringen. Die Funktionsauswertung der Testprobleme benötigt wenig Rechenzeit. Um zeitintensive Auswertungen abzubilden, wurden die Testfunktionen um eine Wartezeit erweitert. Diese beträgt zwischen zwei und vier Sekunden, wodurch zusätzlich zeitliche Unterschiede zwischen den Auswertungen simuliert werden. Die zufällige Generierung der Wartezeit geschieht unabhängig von den Entscheidungsvariablen des Individuums. Das entspricht im Vergleich zur Optimierung der Pfadplanung nicht der Realität, da die benötigte Simulationszeit maßgeblich von den Entscheidungsvariablen abhängt und dazu eine der Zielfunktionen darstellt. Dennoch sollte die zufällige Verlängerung der Auswertung eine Tendenz im Bezug auf die Zeitersparnis erkennen lassen.

Es ist zu erwarten, dass die Ausführungsdauer der Algorithmen sich in etwa im selben Rahmen bewegt; allerdings mit leichten Vorteilen für die asynchrone Variante, da die parallele Ausführung nur so schnell wie der langsamste SMS-EMOA ist. Mit Blick auf die Güte der Lösungen ist zu erwarten, dass die asynchrone Variante die parallel ausgeführten SMS-EMOAs deutlich übertrifft, da von vornherein eine Population optimiert wird. Es profitieren direkt alle, in diesem Fall fünf, Threads/Prozesse von einer Verbesserung. Außerdem verteilt der S-Metrik-Indikator bei der Selektion der nächsten Generation durch die höhere Populationsgröße die Individuen gleichmäßiger auf der Pareto-Front.

Ergebnisse ZDT1

Die Abbildung 4.1 vergleicht die Pareto-Fronten der asynchronen Variante mit der parallelisierten Ausführung mehrerer SMS-EMOAs pro Lauf. Sie zeigt den Beitrag der Algorithmen zu einer gemeinsamen Menge von nicht-dominierten Lösungen. Dazu werden die Lösungsmengen vereinigt und die dominierten Lösungen verworfen. Die gesuchte Menge Q ergibt sich aus den Lösungen der asynchronen Variante A und den Lösungen der parallelisierten SMS-EMOAs P wie folgt:

$$Q = ND(A \cup P)$$

Auf der X-Achse ist der Index des Laufs aufgetragen, während die Y-Achse den Beitrag zu Q des jeweiligen Algorithmus aufzeigt. Die türkisfarbenen Balken stellen die Anzahl der aus A nach Q übernommenen Lösungen des asynchronen SMS-EMOA dar ($|A \cap Q|$). Die Beiträge der parallelisierten SMS-EMOAs ($|P \cap Q|$) bilden die rosafarbenen Balken ab. Unter Berücksichtigung der Populationsgröße von 40 Individuen ist zu sehen, dass in jedem Lauf alle Lösungen der asynchronen Variante in die Menge gemeinsam nicht-dominierter

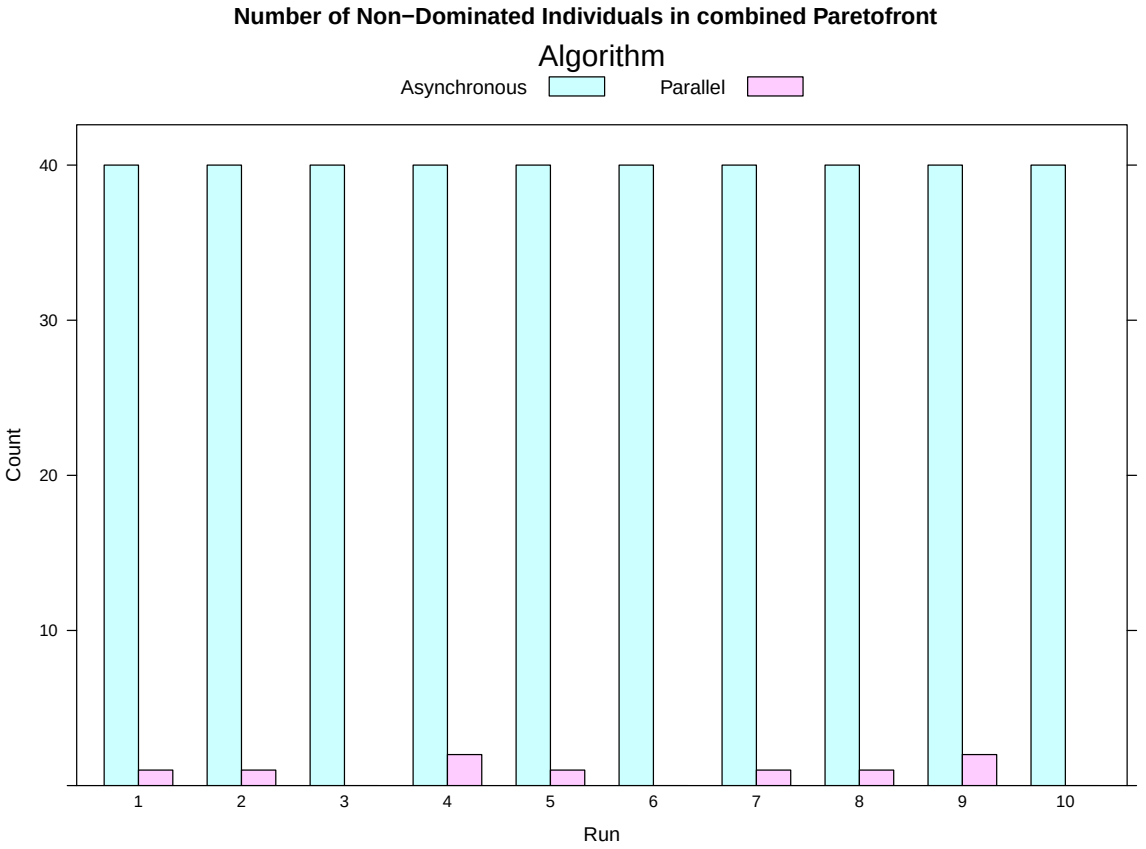


Abbildung 4.1: Vergleich des Beitrags des asynchronen SMS-EMOA und der parallel ausgeführten SMS-EMOAs zu einer gemeinsamen Menge nicht-dominierter Punkte auf ZDT1.

Lösungen aufgenommen werden; $(A \cap Q) = A$. Das bedeutet, dass keine Lösung der parallelisierten SMS-EMOAs eine Lösung der asynchronen Variante dominiert. Dahingegen dominiert die Lösungsmenge des asynchronen SMS-EMOA die meisten Lösungen der parallelisierten Ausführung. Das hat zur Folge, dass nur wenige Lösungen aus P zu Q gehören. In den Läufen 3, 6 und 10 werden alle Lösungen der parallelisierten Ausführung von den Lösungen der asynchronen Variante dominiert ($P \cap Q = \emptyset$).

Diese Dominanz im zweidimensionalen Lösungsraum veranschaulicht Abbildung 4.2. Sie zeigt die Lösungen des repräsentativen Laufs 2. In der Grafik sind die Werte der Zielfunktion f_1 auf der X- und f_2 auf der Y-Achse aufgezeichnet. Im linken Bild, unter *all*, sind die Lösungen der Algorithmen eingezeichnet. Lösungen der asynchronen Variante A sind als blaue Punkte eingezeichnet. Die magentafarbenen Punkte sind Lösungen der parallel ausgeführten SMS-EMOAs. Die echte Pareto-Front ist durch eine graue Kurve dargestellt. Die Lösungen des asynchronen SMS-EMOA decken diese gut ab, während die Lösungen der parallelisierten Ausführung deutlich entfernter liegen. Das rechte Bild, *non-dominated*, zeigt die nicht-dominierten Lösungen der vereinigten Lösungsmenge Q. Die Farbgebung der Punkte lässt erkennen aus welcher Lösungsmenge die eingezeichnete Lösung stammt. Es fällt auf, dass einzig Ausreißer der parallelen Variante nicht von Lösungen der asyn-

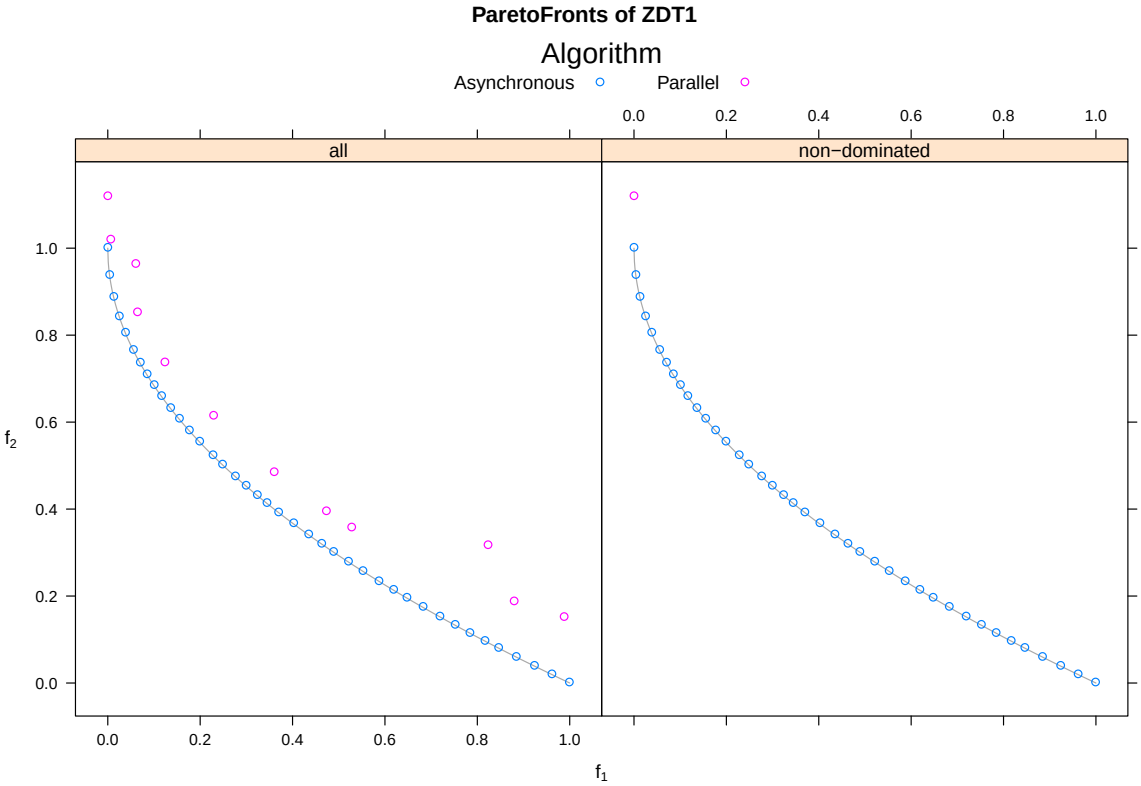


Abbildung 4.2: Vergleich der nicht-dominierten Front des asynchronen SMS-EMOA und parallel ausgeführten SMS-EMOAs. Die graue Linie zeigt die Menge der pareto-optimalen Lösungen.

chronen Variante dominiert werden. Lediglich die parallele Lösung, die eine Zielfunktion sehr stark optimiert, wird in Q übernommen. Es fällt weiterhin auf, dass die asynchrone Lösungsmenge die Pareto-Front nicht nur besser erreicht, sondern auch gleichmäßiger abdeckt.

Die Indikatoren belegen diese visuellen Eindrücke. Die Tabelle 4.2 listet die Werte der einzelnen Indikatoren über zehn Durchläufe gemittelt auf. Der Mittelwert ist durch μ und die Standardabweichung in σ beschrieben. Die Daten, aus denen sie ermittelt wurden, sind in Anhang A.2 in Tabelle A.1 zu finden. Zur Berechnung des Hypervolumenindikators liegt der Referenzpunkt in $(2, 2)^\intercal$. Damit wird er von allen Punkten der Pareto-Front stark dominiert. Das hat zur Folge, dass die äußeren Punkte der Pareto-Front einen positiven Beitrag zum dominierten Hypervolumen leisten.

Die Werte zeigen, dass der asynchrone SMS-EMOA die parallelisierte Ausführung in jedem Indikator übertrifft. Auch im Bezug auf die Ausführungszeit gewinnt der asynchrone SMS-EMOA den Vergleich. Die Zeitersparnis fällt, wie zu erwarten war, verhältnismäßig gering aus. Die Werte und Grafiken zeigen, dass die asynchrone Variante des SMS-EMOA der parallelisierten Ausführung in der Güte der Lösungen deutlich überlegen ist.

Indikator		Asynchron	Parallel
Hypervolumen	μ	3.6522575	3.2535555
	σ	0.000868247574140077	0.121530838965466
Epsilon	μ	0.0160869	0.1975059
	σ	0.00059241141953882	0.045880077481299
Spread	μ	0.1350374	0.5587573
	σ	0.00956413507014618	0.0945228872676348
GD	μ	0.0001896	0.0525876
	σ	4.81231752900824E-05	0.0220560788001857
IGD	μ	0.0003494	0.0046907
	σ	3.07245829914744E-06	0.00127028879000013
Dauer (s)	μ	6003.2	6020.8
	σ	11.7456374880208	8.34026378479722

Tabelle 4.2: Die Tabelle vergleicht die asynchrone Variante des SMS-EMOA mit parallel arbeitenden SMS-EMOAs anhand unterschiedlicher Merkmale auf dem Testproblem ZDT1.

Ergebnisse ZDT3

Die Abbildung 4.3 vergleicht die Pareto-Fronten der asynchronen Variante mit der parallelisierten Ausführung mehrerer SMS-EMOAs pro Lauf. Sie zeigt, wie zuvor Abbildung 4.1, den Beitrag der Algorithmen zu einer gemeinsamen Menge von nicht-dominierten Lösungen. Es ist zu sehen, dass wie im Fall von ZDT1 in jedem Lauf keine Lösung des asynchronen SMS-EMOA von einer Lösung der parallelisierten Variante dominiert wird. Im Vergleich zu den Tests auf ZDT1 bleibt die Anzahl der insgesamt nicht-dominierten Lösungen der parallelen SMS-EMOAs gleich. Sowohl bei der Optimierung von ZDT1 als auch bei der von ZDT3 blieben während der zehn Durchläufe neun Lösungen der parallelisierten Variante nicht-dominiert.

Die Dominanz des asynchronen SMS-EMOA bei der Optimierung von ZDT3 veranschaulicht Abbildung 4.4. Sie zeigt die Lösungen des repräsentativen Laufs 1. Die Grafik ist wie Abbildung 4.11 aufgebaut. Im linken Abschnitt sind die Lösungsmengen beider Algorithmen eingezeichnet, wobei der rechte Abschnitt ausschließlich die nicht-dominierten Lösungen zeigt. Die unzusammenhängende Pareto-Front von ZDT3 ist durch die grauen Linien kenntlich gemacht worden. Die Lösungen des asynchronen SMS-EMOA decken diese gut ab, wenn auch nicht alle Teile der Pareto-Front gleich stark bedeckt werden. So sind die für f_1 optimierten Teile dichter abgedeckt als die in f_2 verbesserten Teilfronten. Es gibt Ausnahmefälle, in denen der in f_2 optimale Teil der Front nicht erreicht wird. Wie schon in

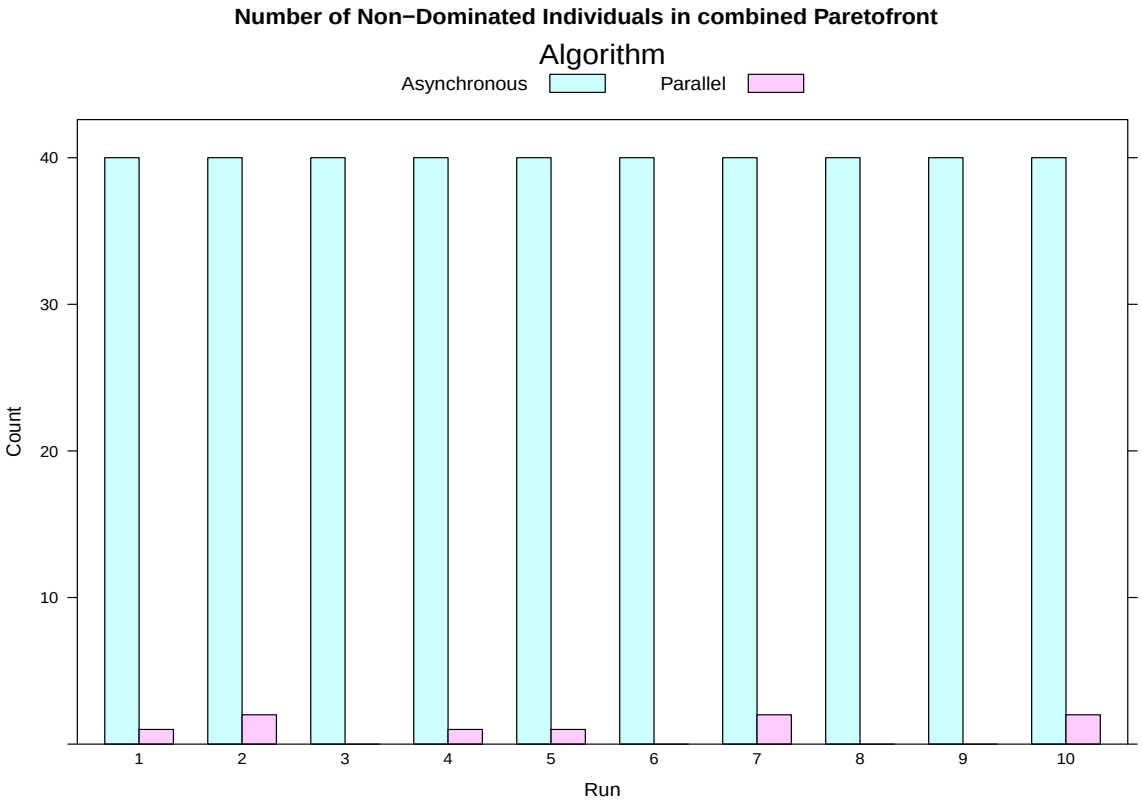


Abbildung 4.3: Vergleich des Beitrags des asynchronen SMS-EMOA und der parallel ausgeführten SMS-EMOAs zu einer gemeinsamen Menge nicht-dominierter Punkte auf ZDT3.

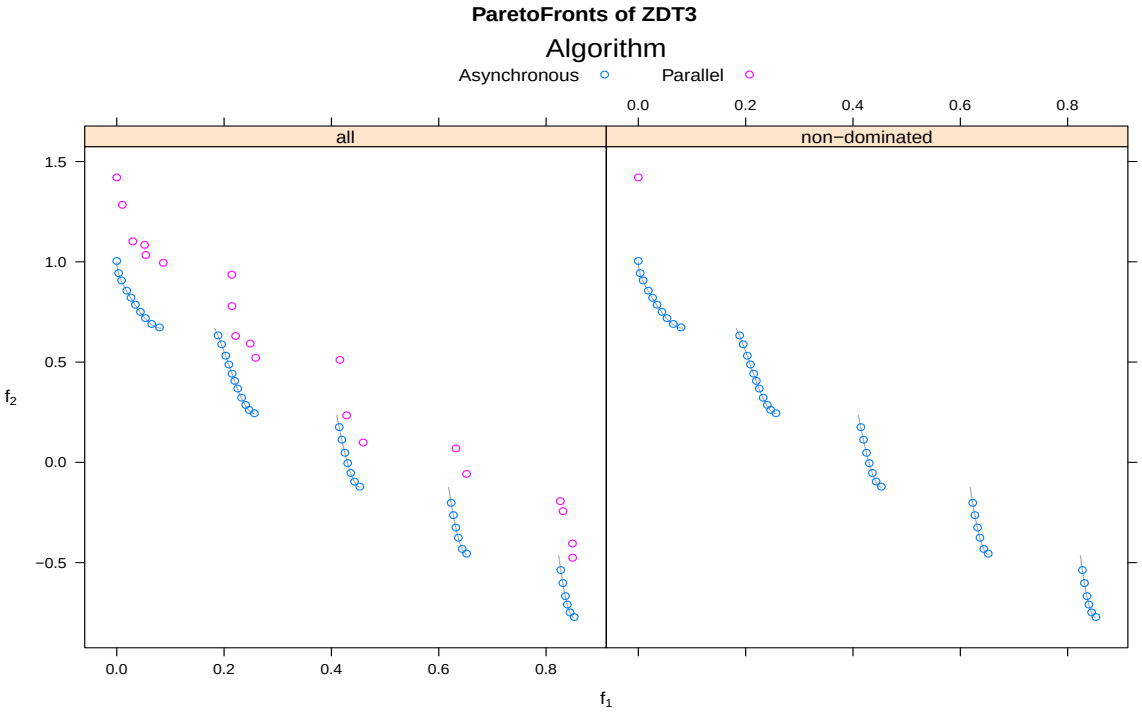


Abbildung 4.4: Vergleich der nicht-dominierten Front des asynchronen SMS-EMOA und parallel ausgeführten SMS-EMOAs. Die grauen Linien zeigen die Menge der pareto-optimalen Lösungen.

Indikator		Asynchron	Parallel
Hypervolumen	μ	4.7341526	4.2058197
	σ	0.14615795771849	0.230468295710299
Epsilon	μ	0.0713905	0.3105196
	σ	0.122901972704469	0.153055959575052
Spread	μ	0.6837661	0.7128661
	σ	0.0105479023549709	0.0696547872402895
GD	μ	0.0001634	0.0323241
	σ	2.1689628858051E-05	0.0102453281201726
IGD	μ	0.0007339	0.0040043
	σ	0.000910576460271184	0.0012393102154021
Dauer (s)	μ	6003.2	6031.7
	σ	12.9135587658863	17.2571724219236

Tabelle 4.3: Die Tabelle vergleicht die asynchrone Variante des SMS-EMOA mit parallel arbeitenden SMS-EMOAs anhand unterschiedlicher Merkmale auf dem Testproblem ZDT3.

ZDT1 werden einzig Ausreißer der parallelen SMS-EMOAs, die ganz besonders stark eine Zielfunktion, in diesem Fall f_1, optimieren, nicht von Lösungen der asynchronen Variante dominiert.

Tabelle 4.3 listet die Werte der einzelnen Indikatoren gemittelt über zehn Durchläufe auf. Die den Werten zu Grunde liegenden Daten sind in Anhang A.2 in Tabelle A.2 aufgelistet. Der Referenzpunkt zur Berechnung des Hypervolumenindikators ist $(2, 2)^\intercal$. Damit wird er von allen Punkten der Pareto-Front stark dominiert. Das hat zur Folge, dass die äußeren Punkte der Pareto-Front einen positiven Beitrag zum dominierten Hypervolumen leisten.

Die Werte bestätigen die zuvor gewonnen Eindrücke. Der asynchrone SMS-EMOA übertrifft die parallelisierte Ausführung in allen Indikatoren. Die zur Optimierung benötigte Zeit ist ebenfalls geringer als die der simultan ausgeführten SMS-EMOAs. Die asynchrone Variante übertrifft die parallel optimierenden SMS-EMOAs damit sowohl in der Güte der Lösungen als auch in der Laufzeit.

Ergebnisse DTLZ1

Die Abbildung 4.5 zeigt den bereits aus den Abbildungen 4.1 und 4.3 bekannten Vergleich von nicht-dominierten Lösungen. Bei der Optimierung dieses Testproblems dominiert erneut keine Lösung der parallelen SMS-EMOAs (pSMS-EMOA-Lösung) eine Lösung der asynchronen SMS-EMOA-Variante (aSMS-EMOA-Lösung). Im Vergleich zu den

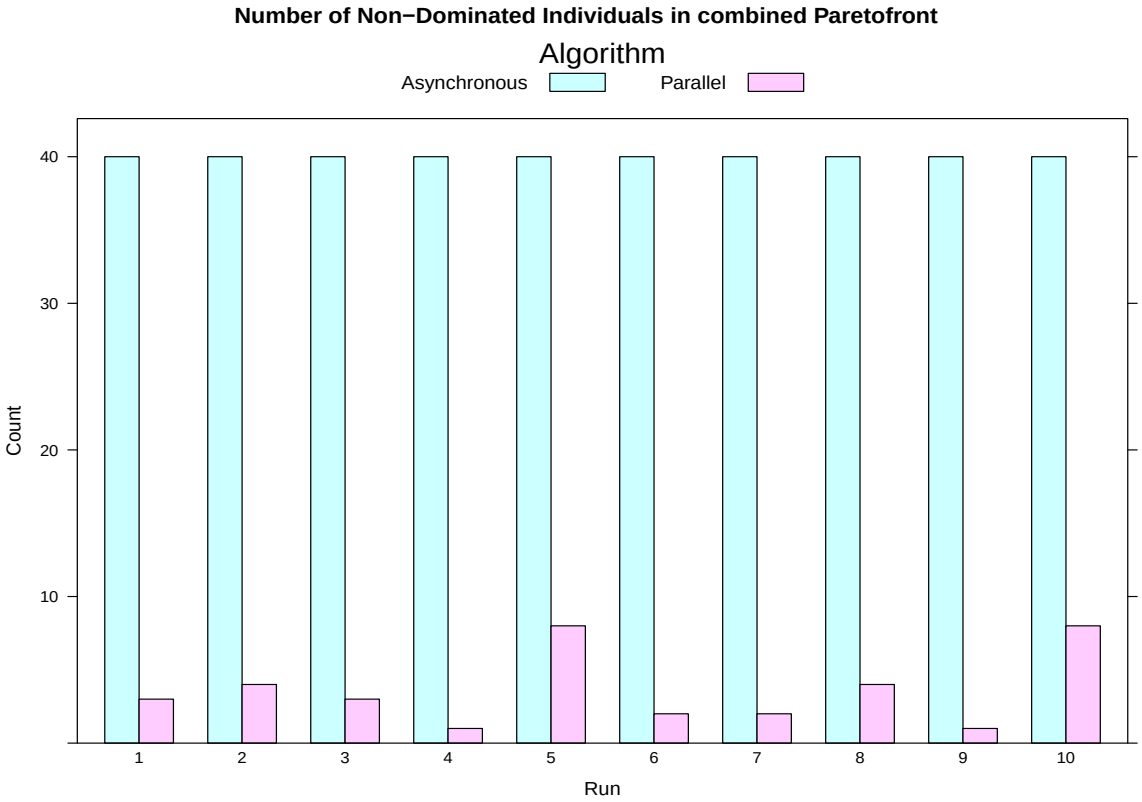


Abbildung 4.5: Vergleich des Beitrags des asynchronen SMS-EMOA und der parallel ausgeführten SMS-EMOAs zu einer gemeinsamen Menge nicht-dominierter Punkte auf DTLZ1.

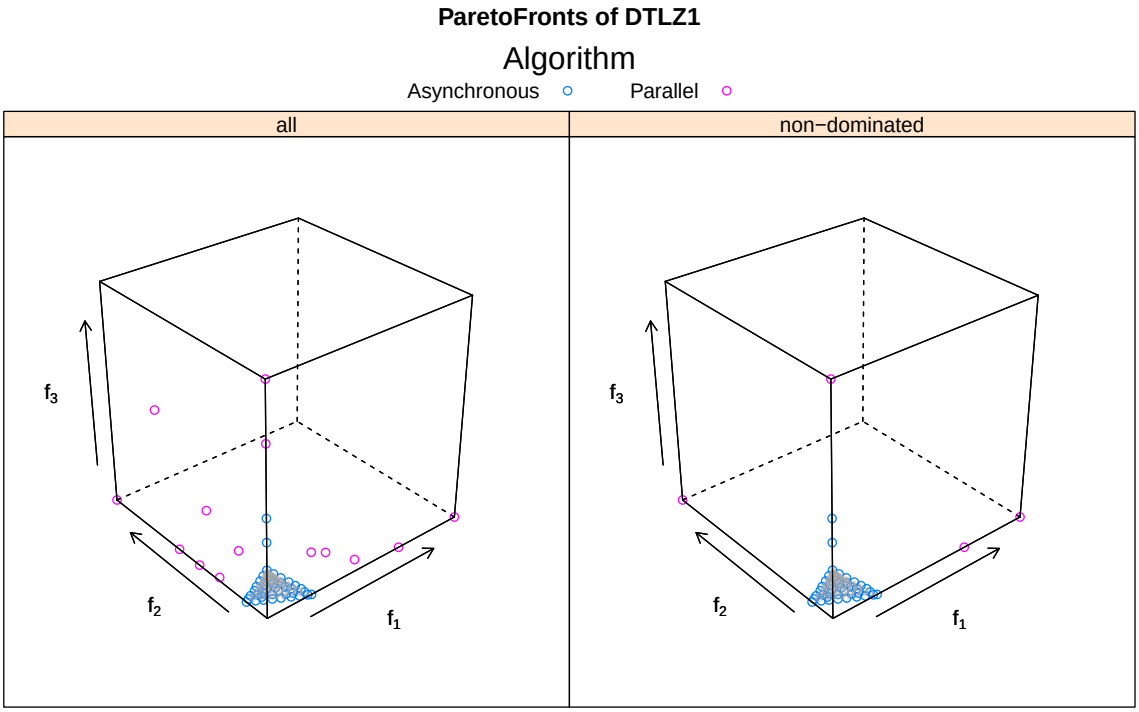


Abbildung 4.6: Vergleich der nicht-dominierten Front des asynchronen SMS-EMOA und parallel ausgeführten SMS-EMOAs. Die grauen Punkte zeigen die Menge der pareto-optimalen Lösungen.

zuvor analysierten Experimenten ist die Anzahl der pSMS-EMOA-Lösungen, die nicht von aSMS-EMOA-Lösungen dominiert werden, gestiegen.

Die Abbildung 4.6 zeigt die Lage der Lösungen im dreidimensinalen Lösungsraum. Die eingezeichneten Lösungen stammen aus dem 2. Lauf. Die Farbgebung der Punkte entspricht der Farbgebung der Grafiken 4.2 und 4.4. Die Aufteilung in die Teilbilder *all* und *non-dominated* ist ebenfalls identisch. Der Unterschied zu den vorherigen Graphen liegt in der zusätzlichen Dimension des Lösungsraums. Die Achsen des Quaders stellen die Zielfunktionen f_1, f_2 und f_3 dar. Die echte Pareto-Front des Testproblems ist die dreieckige graue Fläche. Sie liegt innerhalb der blauen Punktewolke. Dies zeigt, dass die Lösungen des asynchronen SMS-EMOA die echte Pareto-Front gut abdecken. Nicht dominierte pSMS-EMOA-Lösungen sind wie bei den vorherigen Experimenten nur durch Ausreißer in einzelnen Dimensionen in der gemeinsamen nicht-dominierten Menge vertreten (siehe *non-dominated*). Das trifft auf die magentafarbenen Punkte links, rechts und mittig oben zu. Der linke Punkt stellt eine Lösung mit guten Zielfunktionswerten f_1 und f_3 dar. Die beiden rechten Punkte sind Lösungen mit guten Zielfunktionswerten in f_2 und f_3. Lösungen mit niedrigen Zielfunktionswerten in f_1 und f_2 liegen weit vorne im Bild. Das trifft auf den magentafarbenen Punkt mittig oben im Quader zu.

Durch die gestiegene Anzahl von Zielfunktionen sind Ausreißer in einer zusätzlichen Dimension möglich. Das erklärt die gestiegene Anzahl von pSMS-EMOA-Lösungen, die nicht

Indikator		Asynchron	Parallel
Hypervolumen	μ	383.9118723	370.5323579
	σ	0.0914830323218965	15.8226276446003
Epsilon	μ	0.1372992	1.4238958
	σ	0.104067348545834	0.653941256080667
Spread	μ	0.7182646	0.749592
	σ	0.100552841970976	0.156431511069222
GD	μ	0.0993405	1.9983456
	σ	0.0650099169131141	0.793517728083777
IGD	μ	0.0022236	0.0327497
	σ	0.00223058988610636	0.0204613520425704
Dauer (s)	μ	6012.7	6040.9
	σ	12.2151545221499	16.6279884532074

Tabelle 4.4: Die Tabelle vergleicht die asynchrone Variante des SMS-EMOA mit parallel arbeitenden SMS-EMOAs anhand unterschiedlicher Merkmale auf dem Testproblem DTLZ1.

von aSMS-EMOA-Lösung dominiert werden im Vergleich zu den vorherigen Experimenten auf ZDT1 und ZDT3.

Die Tabelle 4.4 stellt die bekannten Indikatoren für DTLZ1 gegenüber. Die Indikatorwerte sind über zehn Durchläufe gemittelt. Die zu Grunde liegenden Daten stehen in Tabelle A.3 in Anhang A.2. Der Referenzpunkt zur Berechnung des Hypervolumens ist so gewählt, dass er von allen Lösungen stark dominiert wird. Damit leisten alle Lösungen einen positiven Beitrag zum dominierten Hypervolumen. In diesem Fall liegt der Referenzpunkt bei $(8, 16, 3)^\intercal$

Alle Indikatoren bewerten die Güte der aSMS-EMOA-Lösungen besser als die Güte der Lösungen der parallelen SMS-EMOAs. Beim Hypervolumen der pSMS-EMOA-Lösungen fällt die vergleichsweise hohe Standardabweichung auf. Diese hängt mit einem schlechten letzten Testlauf zusammen, wie die Daten in Tabelle A.3 zeigen. Die Laufzeit der parallel ausgeführten SMS-EMOAs ist ebenfalls schlechter als die Laufzeit der asynchronen Variante. Damit übertrifft die asynchrone Variante des SMS-EMOA die parallel ausgeführten SMS-EMOAs erneut.

Ergebnisse DTLZ7

Beim Vergleich der nicht-dominierten Lösungen ist, wie Abbildung 4.7 zeigt, der Anteil der pSMS-EMOA-Lösungen weiter gestiegen. Dennoch gibt es auch weiterhin keine Lösung der parallelen SMS-EMOAs, die eine aSMS-EMOA-Lösung dominiert.

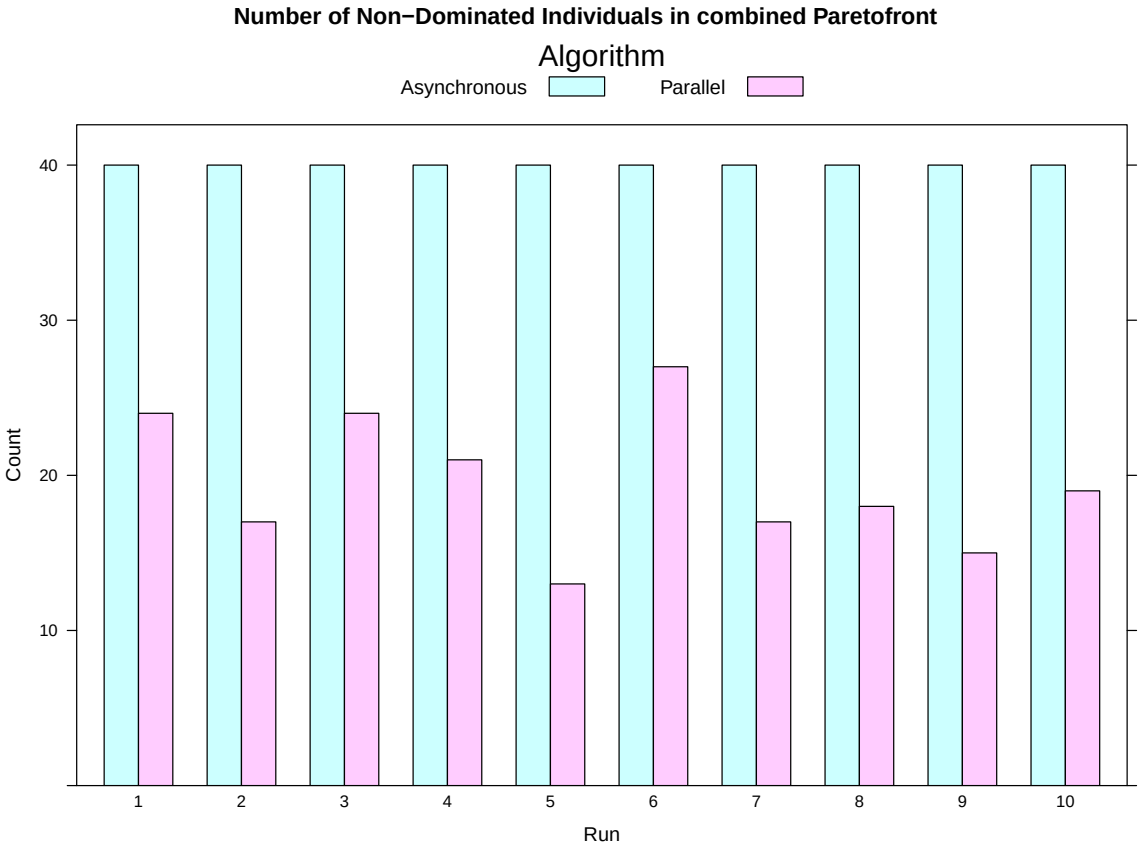


Abbildung 4.7: Vergleich des Beitrags des asynchronen SMS-EMOA und der parallel ausgeführten SMS-EMOAs zu einer gemeinsamen Menge nicht-dominierter Punkte auf DTLZ7.

Wo die Lösungen der beiden Algorithmen im Lösungsraum liegen, zeigen die Abbildungen 4.8 und 4.9. Beide Grafiken visualisieren die Ergebnisse des 8. Laufs. Abbildung 4.8 zeigt die Ergebnisse als Punkte im dreidimensionalen Lösungsraum. Sie ist identisch zu Abbildung 4.6 aufgebaut. Sie veranschaulicht die unzusammenhängende Pareto-Front als graue Punktewolken. Abbildung 4.9 zeigt die gleichen Punkte als Parallele Koordinaten. Die horizontalen grauen Linien sind die Achsen des Koordinatensystems. Jede vertikale Linie stellt einen Punkt im dreidimensionalen Lösungsraum dar. Blaue Linien zeigen aSMS-EMOA-Lösungen und magentafarbene Linien sind pSMS-EMOA-Lösungen. Die Position einer Linie auf der i-ten Achse entspricht der i-ten Koordinate des Punktes. Punkte mit geringem Wert von f_1 sind damit weit links auf der unteren Achse zu finden. Die blauen Linien, die dort starten, nach rechts zur mittleren Achse f_2 und von dort nach links zur oberen Achse f_3 laufen, sind als blaue Punkte ganz links in Abbildung 4.9 zu finden.

Wie aus den Abbildungen zu erkennen ist, optimieren die pSMS-EMOA-Lösungen vor allem die Zielfunktion f_2. Die aSMS-EMOA-Lösungen verteilen sich deutlich besser im Raum und auf die einzelnen unzusammenhängenden Teile der Pareto-Front. Es fällt dabei auf, dass die Punkte vorwiegend auf den Rändern der einzelnen Teile liegen. Das hat zur Folge, dass sie die einzelnen Teile der Pareto-Front nicht gut abdecken.

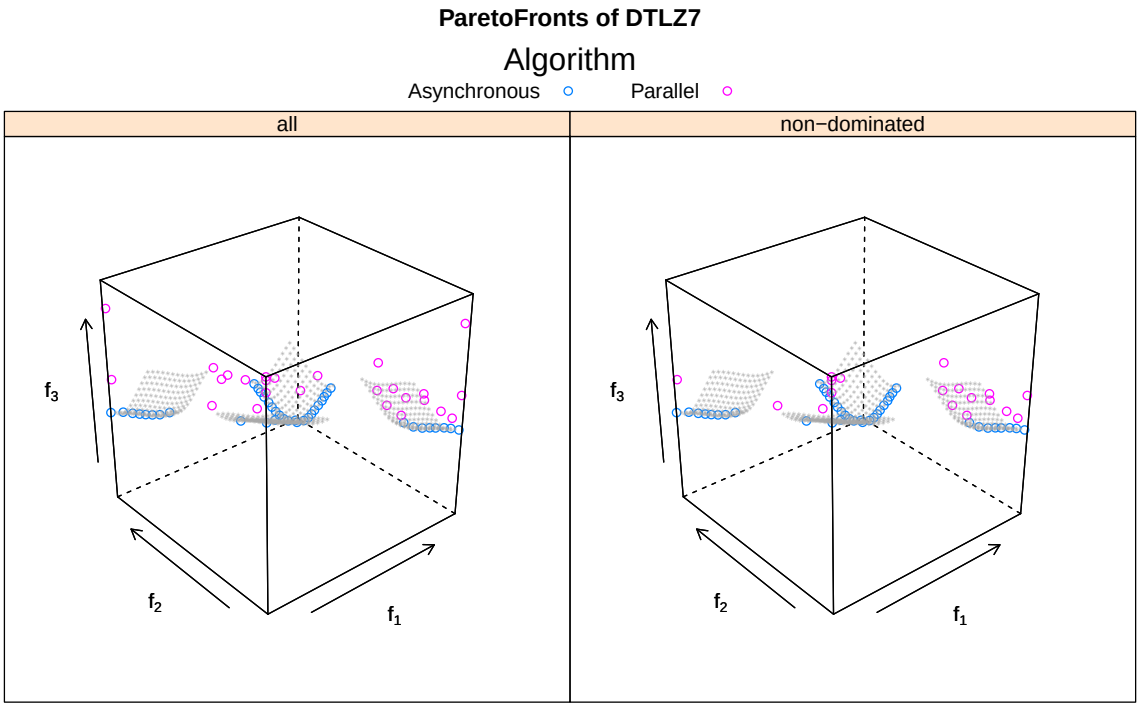


Abbildung 4.8: Vergleich der nicht-dominierten Front des asynchronen SMS-EMOA und parallel ausgeführten SMS-EMOAs. Die Mengen aus grauen Punkten zeigen die unzusammenhängenden Mengen der pareto-optimalen Lösungen.

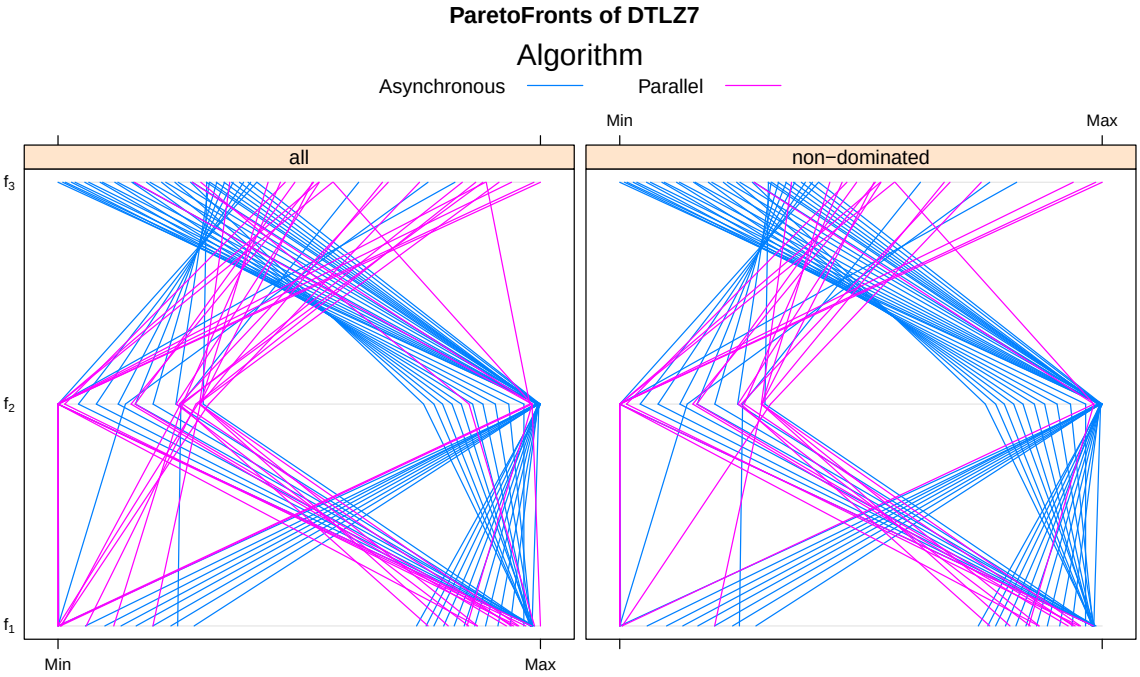


Abbildung 4.9: Vergleich der nicht-dominierten Front des asynchronen SMS-EMOA und parallel arbeitenden SMS-EMOAs auf DTLZ7 in parallelen Koordinaten.

Indikator		Asynchron	Parallel
Hypervolumen	μ	15.3798373	15.076006
	σ	2.36935770770587	0.972054990823461
Epsilon	μ	0.8894538	0.8464264
	σ	0.91739575920317	0.468885407817774
Spread	μ	0.9313174	0.6391409
	σ	0.0766889722948482	0.102025232565724
GD	μ	0.0017955	0.028202
	σ	0.000162848549272015	0.0073815779342902
IGD	μ	0.0126605	0.0084387
	σ	0.00679956315140907	0.0026051565423214
Dauer (s)	μ	6002.7	6037.8
	σ	10.1887192521926	12.5283678106927

Tabelle 4.5: Die Tabelle vergleicht die asynchrone Variante des SMS-EMOA mit parallel arbeitenden SMS-EMOA anhand unterschiedlicher Merkmale auf dem Testproblem DTLZ7.

Besonders die Lösungen der parallelen SMS-EMOAs, welche nicht von den aSMS-EMOA-Lösungen dominiert werden, liegen vorwiegend auf nur einem Teil der Pareto-Front. Es bleiben hauptsächlich die Randbereiche in denen die parallel ausgeführten SMS-EMOAs Lösungen erzeugen können, die nicht von aSMS-EMOA-Lösungen dominiert werden.

Die Tabelle 4.5 vergleicht die Lösungen anhand der bekannten Indikatoren. Die Daten, aus denen die Werte stammen, stehen im Anhang A.2 in Tabelle A.4. Der Referenzpunkt zur Berechnung des dominierten Hypervolumens liegt in $(2, 2, 8)^\intercal$ und wird damit von allen Punkten der Pareto-Front stark dominiert, wodurch alle Punkte einen positiven Beitrag zum dominierten Hypervolumen leisten.

Wie die Indikatoren zeigen, liefert die asynchrone Variante des SMS-EMOA nicht uneingeschränkt bessere Lösungen als die parallelisierten SMS-EMOAs. Besonders beim Hypervolumen- und ϵ-Indikator fallen eine verhältnismäßig hohe Standardabweichung auf, was auf einen oder mehrere Ausreißer in den Optimierdurchläufen schließen lässt. Tatsächlich gibt es beim asynchronen SMS-EMOA in diesen Indikatoren zwei schlechte ($I_{HV} \approx 11.2$, $I_\epsilon \approx 2.5$) und zwei weniger gute ($I_{HV} \approx 14.3$, $I_\epsilon \approx 1.25$) Durchläufe (vgl. Tabelle A.4). Einen der schlechteren Läufe, Lauf 8, zeigt Abbildung 4.10. Wie aus der Grafik zu entnehmen ist, decken die aSMS-EMOA-Lösungen nur einen der vier Teile der echten Pareto-Front ab. Die Lösungen sind somit schlecht über die gesamte Pareto-Front verteilt, wodurch ein niedriger Wert des Hypervolumenindikators entsteht. Der hohe Wert des ϵ-Indikators ist ebenfalls durch die Lokalität der Lösungen zu erklären. Um alle Punkte der

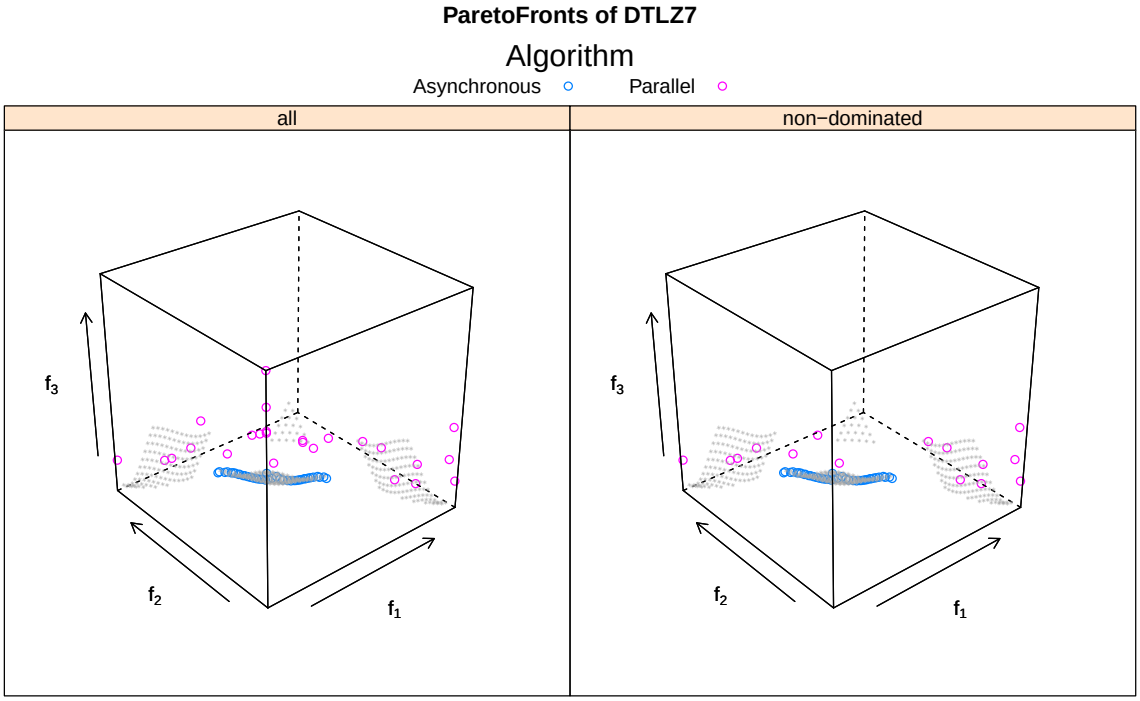


Abbildung 4.10: Nicht-dominierte Front eines schlecht bewerteten Laufs des asynchronen SMS-EMOA. Die Grafik zeigt weiterhin die pSMS-EMOA-Lösungen.

echten Pareto-Front zu dominieren, müssen die aSMS-EMOA-Lösungen weiter verschoben werden als in anderen Fällen. Besonders der Zielfunktionswert f_3 ist in diesem Teil der Pareto-Front im Vergleich zu anderen Teilen recht hoch. Dementsprechend sind für diese Läufe die *IGD*-Indikatorwerte ebenfalls hoch, da die lokalen Lösungen weit von den übrigen Teilen der Pareto-Front entfernt sind. Die weiterhin niedrigen Werte des *GD*-Indikators zeigen an, dass die Lösungen sehr nah an den Werten der Teil-Pareto-Front liegen. Ferner ist die asynchrone Variante den parallelisierten SMS-EMOAs weiterhin in der Laufzeit überlegen.

4.1.3 Vergleich mit unverändertem SMS-EMOA

Im zweiten Teil der Analyse des asynchronen SMS-EMOA tritt dieser gegen einen unveränderten SMS-EMOA an. Die Experimente sollen einen möglichen Leistungsverlust im Bezug auf die Güte der gefundenen Lösungen untersuchen. Dazu verwenden beide Algorithmen die gleichen Operatoren zur Selektion, Rekombination und Mutation. Der asynchrone SMS-EMOA arbeitet wie in Abschnitt 4.1.2 mit fünf Prozessen/Threads gleichzeitig auf der gemeinsamen Population, welche 40 Individuen enthält. Die Populationsgröße der Standardvariante des SMS-EMOA ist identisch groß. Zur Optimierung der jeweiligen Probleme bekommen beide Varianten das gleiche Budget von 10000 Auswertungen. Des Weiteren ist der Offset zur Konstruktion des Referenzpunktes identisch; dieser beträgt 100.

Die Experimente finden auf den gleichen Testproblemen wie zuvor statt. Es ist zu erwarten, dass die Güte der aSMS-EMOA-Lösungen von den Lösungen des unveränderten SMS-EMOA (uSMS-EMOA-Lösungen) übertroffen wird. Durch die simultane Auswertung von bis zu fünf Individuen bei der asynchronen Variante ist damit zu rechnen, dass die Laufzeit der Standardvariante um das fünffache länger ist.

Die Ergebnisse werden von den gleichen Indikatoren wie zuvor bewertet. Allerdings kommt ein neuer Indikator hinzu. Dieser misst die Anzahl von Zielfunktionsauswertungen, die benötigt werden bis das dominierte Hypervolumen der aktuellen Population 98% des Hypervolumens der echten Pareto-Front einnimmt.

Ergebnisse ZDT1

Die Tabelle 4.6 stellt die Werte der Indikatoren gegenüber. Der neu hinzugekommene Indikator der Zielfunktionsauswertungen steht in der Zeile *Evaluationen*. Die Zeile *Non-Dominated* enthält den Beitrag der aSMS-EMOA-Lösungen und uSMS-EMOA-Lösungen an einer gemeinsamen Menge nicht-dominierter Lösungen, gemittelt über zehn Läufe.

Indikator		**Asynchron**	**Standard**
Hypervolumen	μ	3.6516219	3.649516
	σ	0.00106503703691469	0.00879768372925516
Epsilon	μ	0.0155459	0.0170139
	σ	0.000791768078416906	0.00453485965493972
Spread	μ	0.1270656	0.1329461
	σ	0.0166252543451221	0.0467216221089337
GD	μ	0.0002013	0.0001723
	σ	4.5902178597535E-05	2.62223187380521E-05
IGD	μ	0.0003466	0.0003553
	σ	4.00499687890015E-06	3.25178412567624E-05
Non-Dominated	μ	38.6	39.8
	σ	1.8	0.4
Evaluationen	μ	9307.8	9043.1
	σ	382.338802634522	535.757491781496
Dauer (s)	μ	6008.9	29994
	σ	10.6061303028013	39.2198929116335

Tabelle 4.6: Die Tabelle vergleicht die Standardvariante des SMS-EMOA mit der asynchronen Variante anhand unterschiedlicher Merkmale auf dem Testproblem ZDT1.

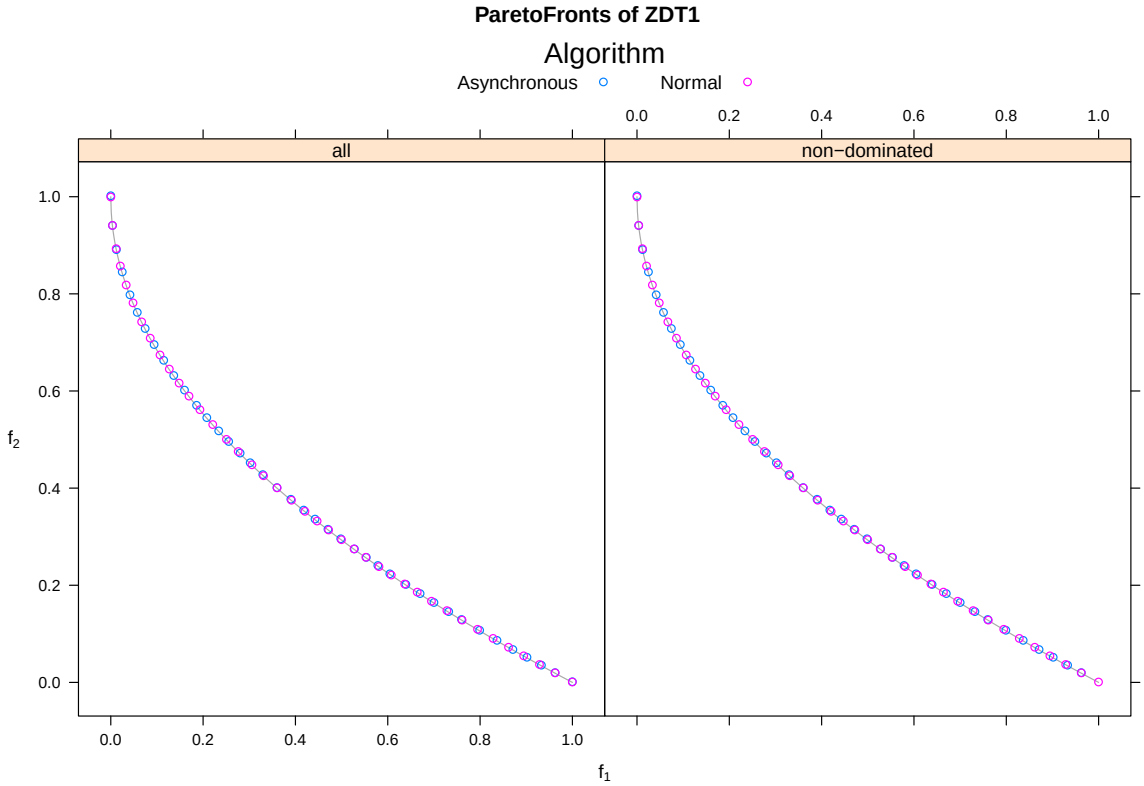


Abbildung 4.11: Vergleich der nicht-dominierten Front des asynchronen SMS-EMOA und der Standardvariante. Die graue Linie zeigt die Menge der pareto-optimalen Lösungen.

Der Wert ersetzt damit die in Abschnitt 4.1.2 verwendeten Balkendiagramme. Zur Berechnung des dominierten Hypervolumens wird der Referenzpunkt aus Abschnitt 4.1.2 $(2, 2)^\intercal$ verwendet.

Die Werte der Indikatoren Hypervolumen I_{HV}, Epsilon I_ϵ, Spread I_Δ, Generational Distance I_{GD} und Inverted Generational Distance I_{IGD} unterscheiden sich nur sehr geringfügig voneinander. Die Güte der aSMS-EMOA-Lösungen und uSMS-EMOA-Lösungen kann daher als gleich angesehen werden. In der Laufzeit ist die asynchrone Variante erwartungsgemäß deutlich besser. Die Anzahl der Zielfunktionsauswertungen ($NFFE$) fällt leicht zugunsten der Standardvariante aus. Diesen Eindruck bestätigt der Non-Dominated-Indikator. Im Schnitt wird eine aSMS-EMOA-Lösung von den uSMS-EMOA-Lösungen dominiert, während nahezu alle Lösungen der uSMS-EMOA-Lösungen nicht-dominiert bleiben. Die Indikatorwerte der einzelnen Testläufe stehen in Tabelle A.5 im Anhang A.2.

Die Verteilung der Lösungen im zweidimensionalen Lösungsraum des ZDT1 zeigt Abbildung 4.11. Die eingezeichneten Lösungen stammen aus dem 3. Lauf. Die Darstellung der Werte entspricht der Abbildung 4.6. Der einzige Unterschied ist, dass die magentafarbenen Punkte hierbei Lösungen des unveränderten SMS-EMOAs sind. Die echte Pareto-Front ist als graue Kurve in beide Teilgrafiken eingezeichnet. Die Lösungen verdecken diese sehr

stark, was bedeutet, dass beide Algorithmen die Pareto-Front erreichen und gleichmäßig abdecken.

Ergebnisse ZDT3

In Tabelle 4.7 sind die Werte der Indikatoren für ZDT3 aufgelistet. Die Werte ergeben sich aus zehn Testläufen, deren Werte in Tabelle A.6 in Anhang A.2 stehen. Zur Berechnung des dominierten Hypervolumens wird der gleiche Referenzpunkt wie in Abschnitt 4.1.2 $(2, 2)^\intercal$ verwendet.

Die Indikatoren bewerten die Algorithmen sehr ähnlich, so dass auf dem Testproblem ZDT3 kein Algorithmus über die Güte seiner Lösungen als signifikant besser bezeichnet werden kann. I_{HV}, I_ϵ und I_{IGD} bewerten den asynchronen SMS-EMOA ein wenig schlechter, während I_Δ und I_{GD} diesen als leicht besser einstufen. Der Non-Dominated-Indikator I_{NDC} und die Anzahl benötigter Evaluationen um 98% des Hypervolumens der Pareto-Front abzudecken I_{NFFE} bewerten den asynchronen ebenfalls etwas besser als den unveränderten SMS-EMOA. Allerdings ist die Standardabweichung in beiden Fällen für die

Indikator		**Asynchron**	**Standard**
Hypervolumen	μ	4.7343517	4.7644493
	σ	0.145916294635692	0.109207170158419
Epsilon	μ	0.0715548	0.0583463
	σ	0.122636553709569	0.100226510296478
Spread	μ	0.6814345	0.6831037
	σ	0.0114795844981428	0.0158569617900151
GD	μ	0.0001826	0.000654
	σ	4.94938380003006E-05	0.00141613918807439
IGD	μ	0.0007311	0.0007034
	σ	0.000911492451970942	0.000854662178875373
Non-Dominated	μ	38.2	37.7
	σ	1.72046505340853	2.3685438564654
Evaluationen	μ	7355.6	7804.8
	σ	318.111992857861	1161.77440150831
Dauer (s)	μ	5999	30042.1
	σ	10.6018866245589	74.476103550065

Tabelle 4.7: Die Tabelle vergleicht die Standardvariante des SMS-EMOA mit der asynchronen Variante anhand unterschiedlicher Merkmale auf dem Testproblem ZDT3.

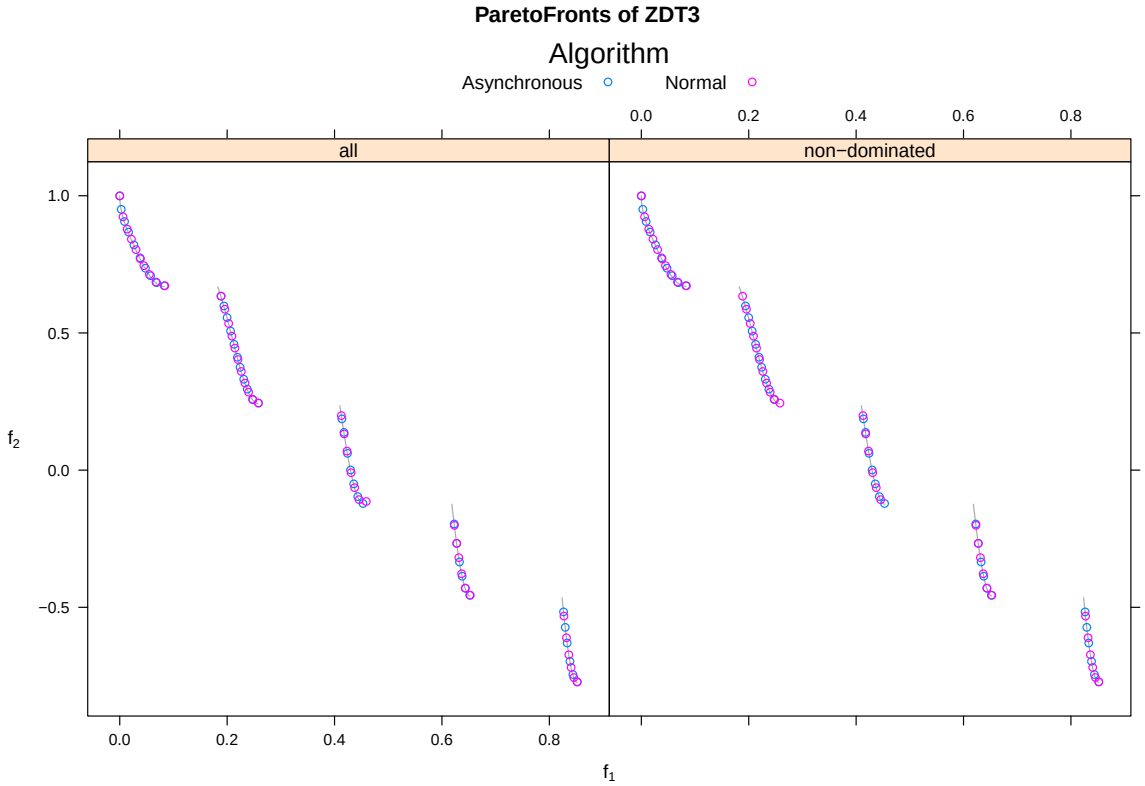


Abbildung 4.12: Vergleich der nicht-dominierten Front des asynchronen SMS-EMOA und der Standardvariante. Die grauen Linien zeigen die Menge der pareto-optimalen Lösungen.

Standardvariante relativ groß, was auf einen Ausreißer in den Testläufen schließen lässt. Wie die Daten in Tabelle A.6 zeigen gibt es für beide Indikatoren jeweils einen negativen Ausreißer. In der Laufzeit I_T liegt die asynchrone Variante erwartungsgemäß deutlich vor dem unveränderten SMS-EMOA.

Die Abbildung 4.12 zeigt die Verteilung der uSMS-EMOA-Lösung und aSMS-EMOA-Lösung auf der in grau eingezeichneten Pareto-Front. Die gezeichneten Werte stammen aus dem repräsentativen Lauf 6. Die Darstellung der Werte ist mit der Abbildung 4.11 identisch. Wie die Abbildung zeigt verteilen beide Algorithmen ihre Lösungen auf allen Teilen der Pareto-Front. Es gibt allerdings Ausnahmefälle in denen der in f_2 optimale Teil der Front nicht erreicht wird.

Ergebnisse DTLZ1

Die Bewertung der Algorithmen durch die bekannten Indikatoren steht in Tabelle 4.8. In der Tabelle A.7 in Anhang A.2 sind die Werte der zehn Testläufe auf DTLZ1 aufgelistet. Als Referenzpunkt des Hypervolumenindikators wird der Referenzpunkt identisch zu Abschnitt 4.1.2 gewählt. Der Punkt $(8, 16, 3)^\intercal$ wird von allen Lösungen stark dominiert, wodurch jede Lösung einen positiven Beitrag zum dominierten Hypervolumen leistet.

Indikator		Asynchron	Standard
Hypervolumen	μ	383.8228682	383.7983859
	σ	0.235873373253451	0.25726701630891
Epsilon	μ	0.1613537	0.2035226
	σ	0.136905332809245	0.190312171950299
Spread	μ	0.5612616	0.6694663
	σ	0.0789397009003708	0.105848475714155
GD	μ	0.0854477	0.1435497
	σ	0.0764231254569584	0.114614412145288
IGD	μ	0.003566	0.0036994
	σ	0.00370048556273363	0.00393228295523097
Non-Dominated	μ	31.2	29.3
	σ	12.7655787177864	14.4710054937451
Evaluationen	μ	n.e.	n.e.
	σ	0	0
Dauer (s)	μ	6009	30074.7
	σ	11.7132403714771	49.087778519709

Tabelle 4.8: Die Tabelle vergleicht die Standardvariante des SMS-EMOA mit der asynchronen Variante anhand unterschiedlicher Merkmale auf dem Testproblem DTLZ1.

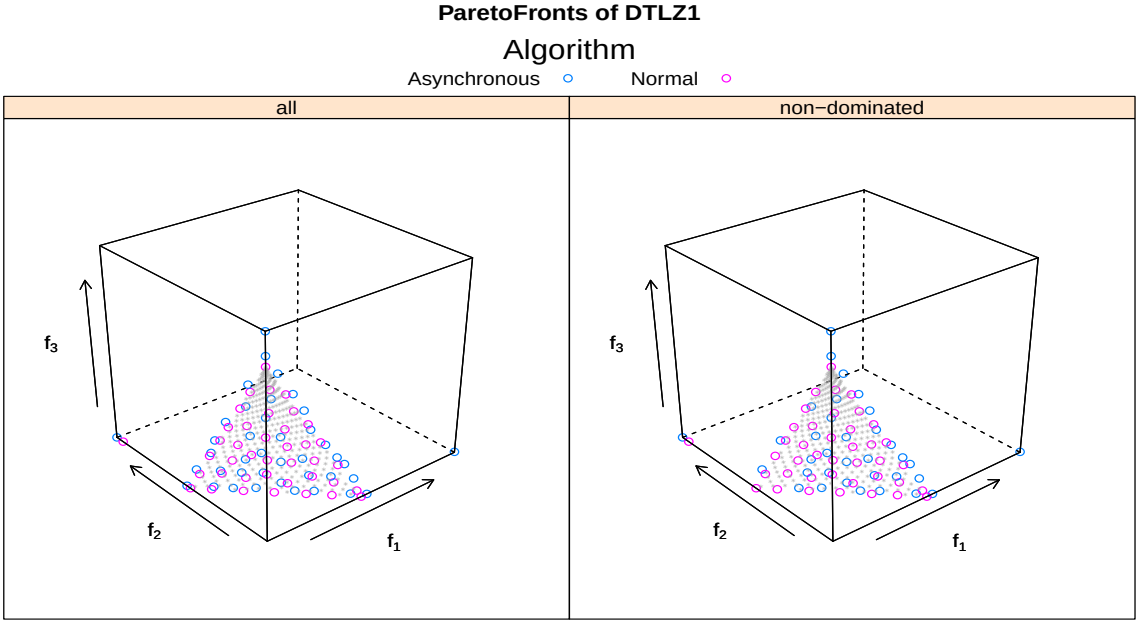


Abbildung 4.13: Vergleich der nicht-dominierten Front des asynchronen SMS-EMOA und der Standardvariante. Die grauen Punkte zeigen die Menge der pareto-optimalen Lösungen.

Wie bei den Experimenten auf ZDT1 und ZDT3 sind die aSMS-EMOA-Lösungen und uSMS-EMOA-Lösungen sehr ähnlich bewertet. Es fällt dabei auf, dass entgegen der angenommenen Hypothese alle Indikatoren den asynchronen SMS-EMOA minimal besser bewerten. Bei I_{NDC} fällt erneut die hohe Standardabweichung auf. Beide Algorithmen haben in diesem Indikator zwei stark negative Ausreißer in ihren Testläufen (vgl. Tabelle A.7). I_{NFFE} zeigt an, dass der erforderliche Schwellwert von 98% von keinem Algorithmus erreicht wird. Die Laufzeiten der beiden Algorithmen liegen im erwarteten Bereich.

Die Abbildung 4.13 visualisiert die aSMS-EMOA-Lösungen und uSMS-EMOA-Lösungen des fünften Testlaufs. Die grauen Punkte stellen die Pareto-Front des Testproblems dar. Die farbigen Punkte visualisieren die Lösungen der jeweiligen Algorithmen. In diesem Experiment liefert der unveränderte SMS-EMOA im Vergleich zur asynchronen Variante die besseren Lösungen. Auf dem linken Schenkel der dreieckigen Pareto-Front werden aSMS-EMOA-Lösungen von uSMS-EMOA-Lösungen dominiert. Diese werden im rechten Bild nicht weiter eingezeichnet. Die Grafik zeigt, dass beide Algorithmen die Pareto-Front erreichen und gleichmäßig abdecken. Sie stellt repräsentative aSMS-EMOA-Lösungen dar.

Ergebnisse DTLZ7

Die Indikatorwerte der Experimente auf DTLZ7 sind in Tabelle 4.9 eingetragen. Die Ergebnisse der einzelnen Tests sind in Tabelle A.8 aufgelistet. Der Referenzpunkt zur Berechnung des dominierten Hypervolumens liegt bei $(2, 2, 8)^\intercal$, womit er alle Lösungen stark dominiert.

Alle Indikatoren bewerten die aSMS-EMOA-Lösungen und pSMS-EMOA-Lösungen sehr ähnlich. Kein Algorithmus kann als überlegen bezeichnet werden, da keine signifikanten Unterschiede in den Indikatoren feststellbar sind. Die Hypothese, dass der unveränderte SMS-EMOA bessere Lösungen generiert als die asynchrone Variante, kann nicht bestätigt werden. Widerlegt ist sie dennoch nicht, da die Ergebnisse der Algorithmen zu ähnlich sind, um bei der geringen Anzahl von Testläufen aussagekräftig zu sein. Dazu werden die Konfidenzintervalle des Hypervolumenindikators der einzelnen Algorithmen betrachtet. Der Standardfehler von I_{HV} beträgt für aSMS-EMOA-Lösungen 0.598 und für uSMS-EMOA-Lösungen 0.436. Damit ergibt sich ein Konfidenzintervall von $[15.09, 17.434]$ für I_{HV} der aSMS-EMOA-Lösungen mit einem Konfidenzniveau von 95% unter der Annahme die Werte für I_{HV} seien normalverteilt. Für uSMS-EMOA-Lösungen resultiert daraus ein Konfidenzintervall von $[14.585, 16.295]$. Da die Konfidenzintervalle nicht disjunkt sind, kann die Hypothese sich weiterhin als wahr herausstellen.

Die Abbildung 4.14 visualisiert die Verteilung der aSMS-EMOA-Lösungen und uSMS-EMOA-Lösungen auf der Pareto-Front, welche als graue Punktewolken eingezeichnet ist. Die Grafik zeigt die Lösungen des zehnten Laufs, welcher repräsentative aSMS-EMOA-Lösungen liefert. Die uSMS-EMOA-Lösungen dieses Testlaufs können allerdings nicht als

Indikator		Asynchron	Standard
Hypervolumen	μ	16.2621014	15.440867
	σ	1.89429076622219	1.38154289742831
Epsilon	μ	0.5449412	0.8449779
	σ	0.734646732757017	0.517219589417193
Spread	μ	0.9486386	0.9403752
	σ	0.0398278867031632	0.0508141727981475
GD	μ	0.0018027	0.0019035
	σ	0.00017521760756271	0.000171761608050228
IGD	μ	0.0100594	0.0129815
	σ	0.00546991521323686	0.00448981365426228
Non-Dominated	μ	39.2	39.4
	σ	0.871779788708135	0.66332495807108
Evaluationen	μ	n.e.	n.e.
	σ	0	0
Dauer (s)	μ	6006.5	30040.3
	σ	6.78601503092942	42.8533545944772

Tabelle 4.9: Die Tabelle vergleicht die Standardvariante des SMS-EMOA mit der asynchronen Variante anhand unterschiedlicher Merkmale auf dem Testproblem DTLZ7.

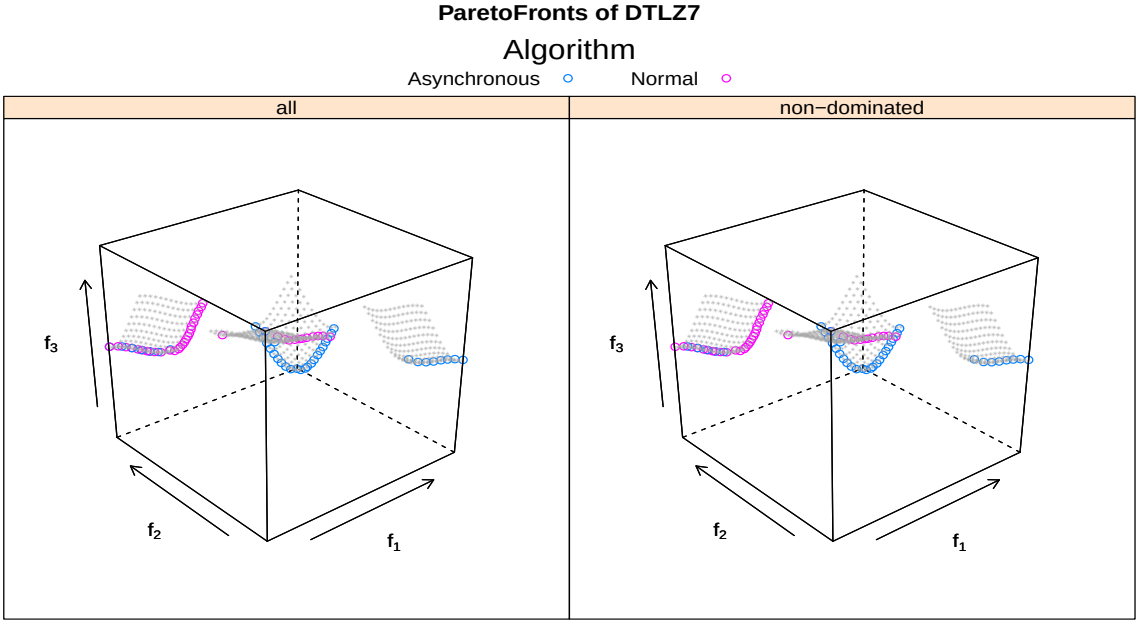


Abbildung 4.14: Vergleich der nicht-dominierten Front des asynchronen SMS-EMOA und der Standardvariante.

repräsentativ angesehen werden. Sie liefern das geringste Hypervolumen der zehn Testläufe und erreichen lediglich zwei der vier Teile der Pareto-Front.

4.2 Analyse der Parameteroptimierung

Die Pfadplanungsparameter des *Adept Lynx* werden in einer Simulation, wie in Abschnitt 3.4.1 beschrieben, optimiert. Die Parameter des Roboters werden sowohl von der asynchronen Variante des SMS-EMOA als auch von parallel ausgeführten SMS-EMOAs optimiert. Um die Ergebnisse vergleichbar zu gestalten, verwenden beide Varianten die gleichen Operatoren zur Selektion, Rekombination und Mutation. Die verwendeten Operatoren sind in Abschnitt 3.1 beschrieben. Darüber hinaus ist der Offset zur Konstruktion des Referenzpunktes identisch.

4.2.1 Analyse der Simulation

Vorüberlegungen Als Vorbereitung zur Optimierung wurden die Parameter gleichverteilt über ihre zulässigen Wertebereiche verändert. Dabei kam heraus, dass besonders die Zielfunktion der Glätte translational über einen großen Wertebereich verfügt. Dadurch können Verbesserungen dieser Zielfunktion einen großen Anteil am dominierten Hypervolumen einnehmen. Um die Randpunkte bei der Selektion der nächsten Generation beizubehalten, ist ein Offset von 100000 gewählt worden.

Setup Der Grad der Parallelisierung beträgt beim folgenden Experiment 16. Er ergibt sich aus den zur Verfügung stehenden Simulatoren. Trotz der zeitintensiven Zielfunktionsauswertungen bekommen die Algorithmen ein hohes Budget von 160000 Auswertungen, welche, wie in Abschnitt 4.1.2 beschrieben, gleichmäßig unter den parallelisierten SMS-EMOAs aufgeteilt werden. Die Population der asynchronen Variante besteht aus 320 Individuen, woraus sich eine Populationsgröße von 20 Individuen bei den parallel ausgeführten SMS-EMOAs ergibt.

Die Optimierung findet auf einer Karte vom Applikationslabor Adept Dortmund statt, welche in Abbildung 4.15 zu sehen ist. Dadurch, dass die Parameter auf einer realen Karte optimiert werden, können die Lösungen anschließend mit dem echten Roboter auf ihr verifiziert werden. Die Karte besteht aus den vier Punkten *Opt_1*, *Opt_2*, *Opt_3* und *Opt_4*, die nacheinander in der Route *Optimization* abgefahren werden. Der Roboter startet auf einer definierten Position rechts oberhalb von *Opt_1*. An diesem Punkt wird der neue Parametersatz für das nächste Experiment eingestellt und die Route gestartet. Bevor der Roboter den ersten Zielpunkt *Opt_1* anfährt, sendet er einen Text mit seiner Systemzeit an das auswertende Programm. Das Senden der Systemzeit ist bereits Teil der Route. Ebenso sendet der Roboter nach erfolgreicher Ankunft an *Opt_4* als letzte Aktion der Route noch einmal seine Systemzeit.

Abbildung 4.15: Karte auf der die Pfadplanungsparameter optimiert werden.

Der Bereich in dem der *Adept Lynx* fährt, ist durch verbotene Zonen eingeschränkt. Der Arbeitsbereich ist so gewählt, dass der Roboter weder sich selbst, noch anderen Personen oder Gegenschäden Schaden zufügen kann. Die kürzeste und schnellste Route ist direkt von *Opt_1* nach oben zu *Opt_2* zu fahren. Diese Strecke verfügt über eine enge Passage zwischen zwei verbotenen Zonen hindurch. Je nach Parametersatz kann es dazu kommen, dass der *Adept Lynx* einen Umweg in Kauf nehmen muss, weil auf Grund der angenommenen Roboterbreite eine Pfadplanung durch die Passage fehlschlägt. In diesem Fall fährt er nicht gerade nach oben, sondern erreicht den zweiten Wegpunkt *Opt_2* nur, indem er außen herum *Opt_4* und *Opt_3* passiert. Diese Wegpunkte gelten selbstverständlich nicht als erreicht, sondern der *Adept Lynx* muss nach Erreichen von *Opt_2* die restlichen Punkte anfahren. Falls ein Fahrzeit-optimierter Parametersatz es dem Roboter nicht ermöglicht, den direkten Weg zu nehmen, ist die Optimierung gescheitert.

Visualisierung und Analyse der Ergebnisse Die Abbildung 4.16 zeigt die Verteilung der Lösungen der parallelen SMS-EMOAs im dreidimensionalem Lösungsraum. Das linke Bild, namens *all*, zeichnet alle gefundenen Lösungen als verschiedenfarbige Punkte. Die Farbgebung gibt Aufschluss, welcher der 16 Algorithmen die Lösung generiert hat. Auf der X-Achse ist der Zielfunktionswert für die translationale Laufruhe als $f_{SmoothTrans}$ eingezeichnet. Die Y-Achse stellt den Wert der Zielfunktion in der rotationalen Glätte als $f_{SmoothRot}$ dar. Die Zeit, die der *Adept Lynx* zum Abfahren der Route benötigt, ist auf der Z-Achse des Koordinatensystems als $f_{Duration}$ aufgetragen.

Die rechte Grafik, namens *non-dominated*, zeichnet ausschließlich die nicht-dominierten Lösungen. Sie stellt damit die bereinigte Lösungsmenge der parallel ausgeführten SMS-

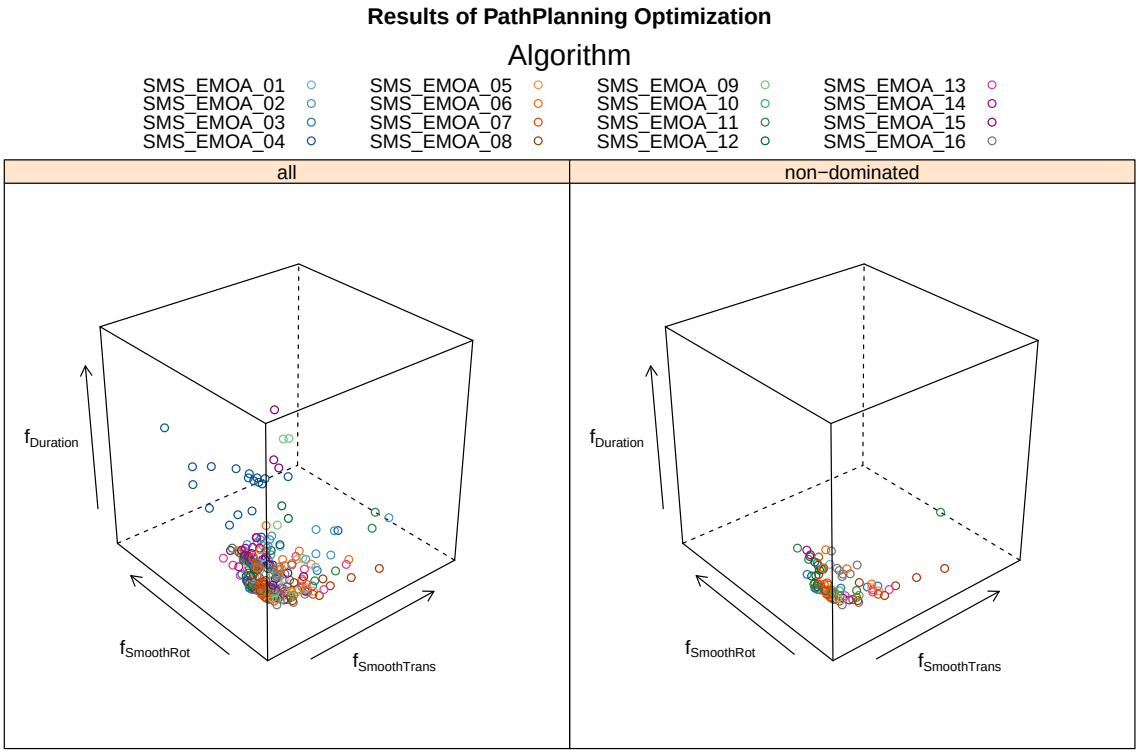


Abbildung 4.16: Vergleich der Lösungen innerhalb der parallel ausgeführten SMS-EMOA.

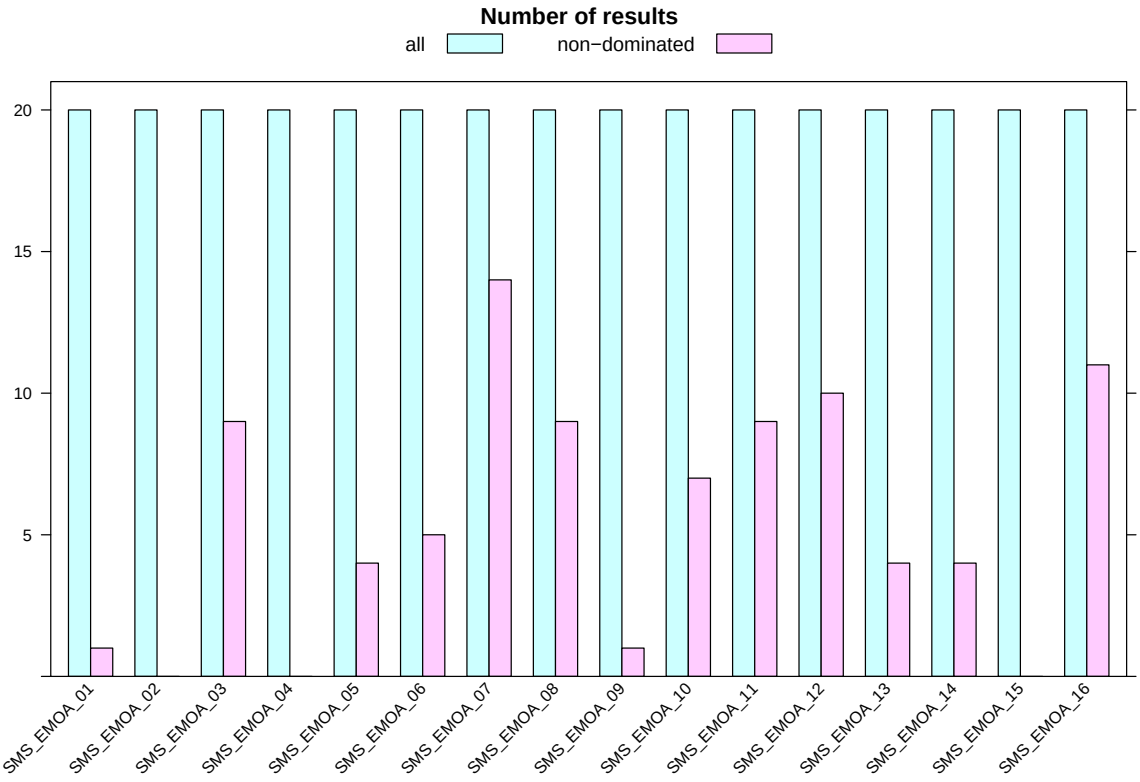


Abbildung 4.17: Visualisierung des Beitrags einzelner paralleler SMS-EMOA zur gemeinsamen Pareto-Front.

Algorithmus	Hypervolumen	Lösungen	I_{NDC}	Laufzeit (h)
Asynchron	17.872E12	318	317	168.62
Parallel	17.261E12	88	9	267.80

Tabelle 4.10: Vergleich von Hypervolumen, Anzahl erzeugter Lösungen und Laufzeit der Lösungen der Pfadplanung.

EMOAs dar. Die Farben der Punkte entsprechen den Farben des linken Bildes. Wie der Vergleich der beiden Abbildungen zeigt, generieren die verschiedenen SMS-EMOAs viele Lösungen, die einander dominieren. Dadurch sind in der Pareto-Front der pSMS-EMOA-Lösungen lediglich 88 von 320 möglichen Lösungen vorhanden. Die Anzahl nicht-dominierter Lösungen in der gemeinsamen Pareto-Front zeigt Abbildung 4.17. Auf der X-Achse sind die einzelnen Algorithmen aufgetragen, während die Y-Achse die Anzahl der Lösungen darstellt. Der türkisfarbene Balken gibt die Anzahl erzeugter Lösungen des jeweiligen Algorithmus an. Der Anteil an der gemeinsamen Pareto-Front ist durch den rosafarbenen Balken gegeben. Die Grafik zeigt, dass manche der 16 Algorithmen keine Lösung generieren, welche nicht von einer anderen Lösung dominiert wird. Diese Algorithmen tragen keine Lösung zur gemeinsamen Pareto-Front der parallelisierten SMS-EMOAs bei.

Die Tabelle 4.10 zeigt Indikatoren zur Bewertung der Lösungen. Es werden die aSMS-EMOA-Lösung mit den pSMS-EMOA-Lösung verglichen. Bei der Berechnung des I_{HV} für die aSMS-EMOA-Lösungen und die pSMS-EMOA-Lösung wird der gleiche Referenzpunkt verwendet. Dieser Punkt muss beide Lösungsmengen stark dominieren. Dazu wird der Referenzpunkt aus den schlechtesten Werten beider Lösungsmengen in den jeweiligen Dimensionen konstruiert. Die Koordinate r_i ergibt sich aus $r_i = 2 \cdot \max(a_i, p_i)$ mit a_i und p_i als schlechtester Wert i-ten Dimension der beiden Pareto-Fronten. Der Indikator zeigt, dass die Qualität der aSMS-EMOA-Lösung die Qualität der pSMS-EMOA-Lösung übertrifft. Die Spalte *Lösungen* stellt die Anzahl generierter Lösungen gegenüber. Der asynchrone SMS-EMOA erzeugt 318 von 320 möglichen Lösungen. Die Population erhält daher zum Ende der Optimierung zwei dominierte Lösungen. I_{NDC} gibt an, wie hoch der Beitrag zu einer gemeinsamen Menge nicht-dominierter Lösungen ist. Die Werte zeigen, dass eine der 318 aSMS-EMOA-Lösungen von Lösungen der parallelen SMS-EMOAs dominiert wird. Es gibt hingegen nur neun pSMS-EMOA-Lösungen, die nicht von Lösungen der asynchronen SMS-EMOA-Variante dominiert werden. Die Laufzeit der asynchronen Variante liegt deutlich unter der Laufzeit der parallel ausgeführten Algorithmen. Sie benötigt zur Optimierung der Parameter 168.62 Stunden und damit circa 100 Stunden weniger als die parallelen SMS-EMOA. Diese Differenz entsteht durch die großen Unterschiede in der Auswertungsdauer einzelner Parametersätze.

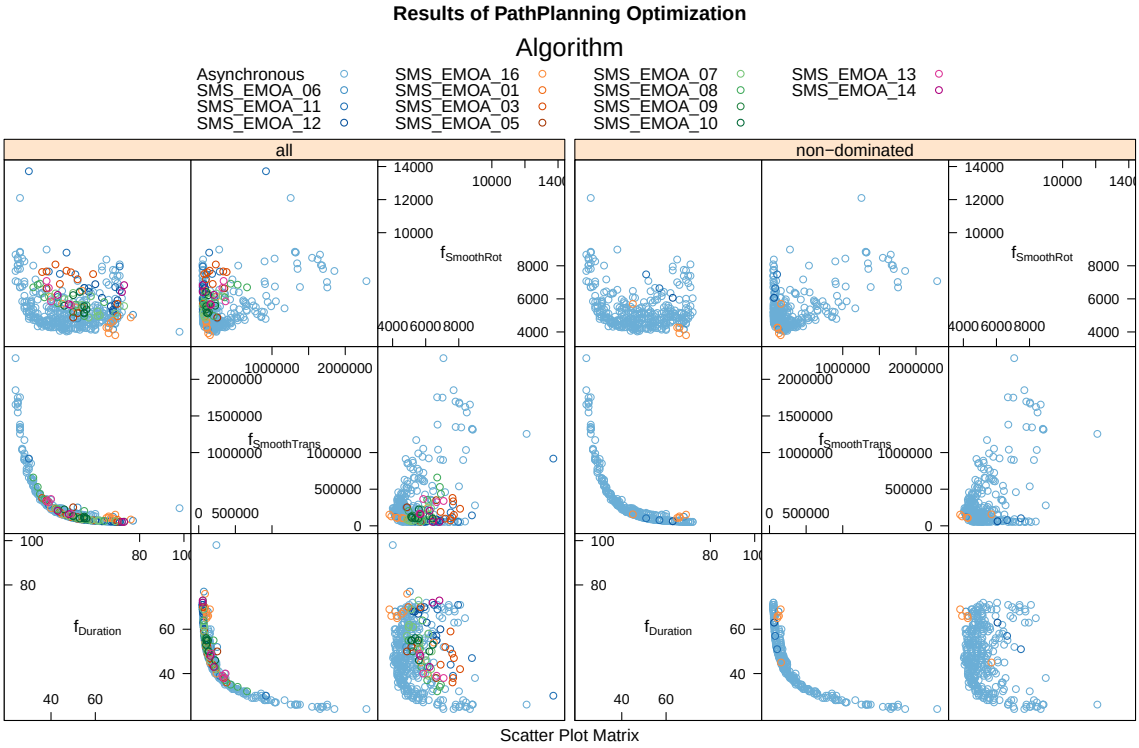


Abbildung 4.18: Vergleich der Lösungen der parallelen und asynchronen Variante für die Pfadplanung als Scatter Plot Matrix.

Wie die Abbildung 4.18 zeigt, sind die Zielfunktionen $f_{Duration}$ und $f_{SmoothTrans}$ konfliktär. Dadurch braucht der Roboter mit in $f_{SmoothTrans}$ guten Parametersätzen länger, um den Parcours abzufahren. Das resultiert in einer verlängerten Evaluationszeit. Diese Parametersätze bleiben über mehrere Generationen erhalten und generieren Nachkommen, die ähnliche Eigenschaften besitzen. Je mehr in $f_{SmoothTrans}$ optimierte Individuen in der Population erhalten sind, desto größer ist die Wahrscheinlichkeit einen Nachkommen mit einer hohen Evaluationszeit zu generieren. Das verlangsamt den Algorithmus und verursacht die großen Unterschiede in der Laufzeit. Der asynchrone SMS-EMOA wird davon nicht so stark beeinflusst, da er durch die größere Population die Pareto-Front gleichmäßiger abdeckt.

Wie weiterhin aus Abbildung 4.18 hervorgeht, ist das Verhältnis zwischen $f_{Duration}$ und $f_{SmoothRot}$ ebenfalls leicht konfliktär. Es ist allerdings bei weitem nicht so extrem ausgeprägt wie zwischen $f_{Duration}$ und $f_{SmoothTrans}$. Der Konflikt zwischen den Zielfunktionen zur Optimierung der Laufruhe und der Optimierung der Laufzeit tritt nicht unerwartet auf. Besonders der Konflikt zwischen $f_{SmoothTrans}$ und $f_{Duration}$ kommt durch das Design des *Adept Lynx* zustande. Er besitzt zwei Freiheitsgrade, mit denen er rotieren und sich vorwärts bewegen kann. Dadurch muss zum Abfahren der Strecke translational beschleunigt werden. Je schneller der Roboter fährt, desto schneller ist er am Ziel aber desto mehr

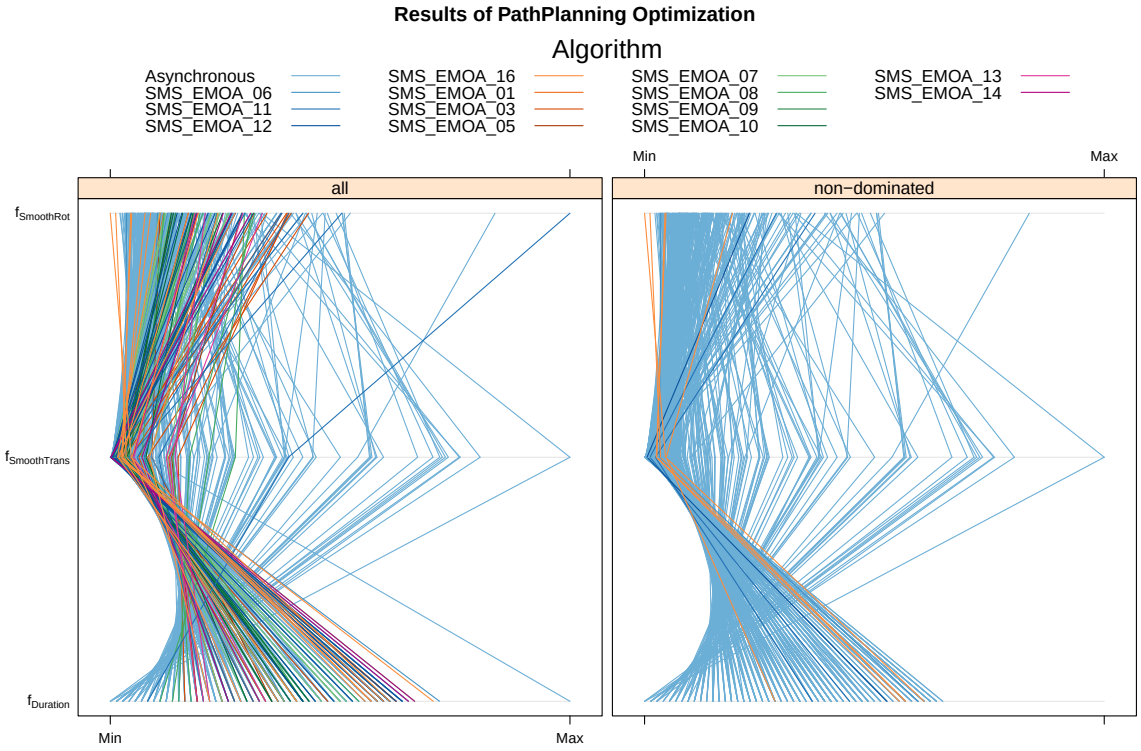


Abbildung 4.19: Vergleich der Lösungen der parallelen und asynchronen Variante für die Pfadplanung als Parallele Koordinaten.

verändert sich seine Geschwindigkeit, was einen höheren Wert in $f_{SmoothTrans}$ zur Folge hat. Zwischen $f_{SmoothTrans}$ und $f_{SmoothRot}$ kann keine Korrelation gefunden werden.

Die Lage der Lösungen im Raum zeigt die Abbildung 4.19. Sie veranschaulicht außerdem, in welchen Bereichen die nicht-dominierten Lösungen der parallelen SMS-EMOAs liegen. Dabei fällt auf, dass sie vor allem in Bereichen liegen, in denen Glätte-Werte optimiert werden. In der Laufzeit liegen sie im höheren Bereich der aSMS-EMOA-Lösungen. An dieser Stelle ist auch eine der beiden dominierten Lösungen der asynchronen SMS-EMOA-Variante zu finden. Sie stellt den schlechtesten Wert für $f_{Duration}$ dar.

Vergleich mit Standardkonfiguration An dieser Stelle soll ein Vergleich zur aktuellen Standardkonfiguration angestellt werden. Die Zielfunktionen $f_{Duration}$ und $f_{SmoothTrans}$ sind aus den oben beschriebenen Gründen konfliktär. Dennoch gibt es Fälle im Fahrverhalten des *Adept Lynx*, die beide Zielfunktionen verbessern oder eine verbessern ohne die andere zu verschlechtern. Ein unnötiger Umweg zum Beispiel bedeutet sowohl eine längere Fahrzeit, als auch mehr Geschwindigkeitsänderungen durch zusätzliches Abbremsen und Beschleunigen in Kurven. Außerdem ist es wünschenswert, bis zur eingestellten Höchstgeschwindigkeit zu Beschleunigen und diese bis zur Zielanfahrt zu halten. Muss durch geschickte Pfadplanung während der Pfadverfolgung nicht abgebremst werden, kommt dies $f_{SmoothTrans}$ durch geringere Geschwindigkeitsänderungen und $f_{Duration}$ durch eine höhere

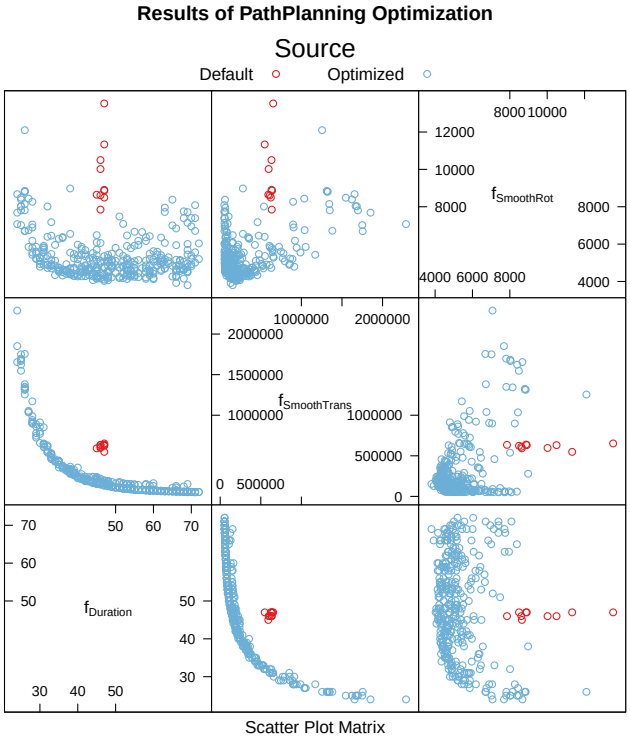


Abbildung 4.20: Vergleich der optimierten Lösungen mit der Standardkonfiguration als Scatter Plot Matrix.

Durchschnittsgeschwindigkeit zugute. Es gibt gerade in diesen Bereichen Optimierungsspielraum, der genutzt werden soll. Dazu ist der simulierte Roboter den Testparcours mit der aktuellen Standardkonfiguration abgefahren.

Die Abbildung 4.20 erweitert die Abbildung 4.18 um die Daten dieser Experimente. Die Resultate der Experimente sind als rote Punkte eingezeichnet. Um eine bessere Vorstellung der räumlichen Distanz zu bekommen, visualisiert Abbildung 4.21 die Daten als dreidimensionale Punktewolke. Die Grafiken zeigen, dass die Optimierung eine Verbesserung in allen Zielfunktionen gebracht hat. Es gibt zusätzlich Lösungen, welche die Standardkonfiguration in einer Zielfunktion verbessern und in einer anderen verschlechtern. Insgesamt dominieren die gefundenen Lösungen die Werte der Standardkonfiguration oder sind mit ihnen unvergleichbar.

4.2.2 Verifikation der Lösungen

Dieser Abschnitt verifiziert die Ergebnisse der Simulation auf dem realen Roboter.

Setup Der echte *Adept Lynx* fährt dazu die gleiche Route wie der virtuelle Roboter beim Optimieren der Parameter in der Simulation. Der Zyklus beginnt in diesem Fall damit, dass der Roboter zum Ausgangspunkt gefahren wird. Anschließend werden die optimierten Parameter gesetzt und die Route gestartet. Mit dem Beenden der Route endet auch der

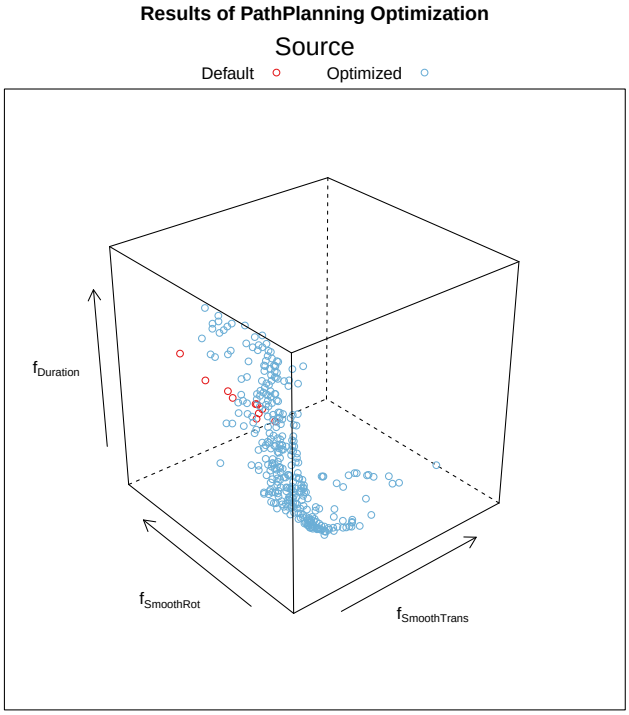


Abbildung 4.21: Vergleich der optimierten Lösungen mit der Standardkonfiguration als 3D-Punktewolke.

Zyklus. Der Test des nächsten Parametersatzes beginnt daher wieder mit der Fahrt zum Ausgangspunkt. Da in der realen Welt mehr Einflüsse auf den Roboter wirken als in der Simulation, wird jeder Parametersatz zehn mal getestet. Der gesamte Testzyklus wird pro Parameter zehn mal durchlaufen. Um das Rauschen zu filtern, wird der Mittelwert dieser zehn Testzyklen als Lösung des Parametersatzes genutzt.

Visualisierung und Analyse der Ergebnisse Die Abbildungen 4.22 und 4.23 zeigen die Ergebnisse der Testläufe im Vergleich zu den Lösungen der Simulation. In den Grafiken sind die optimierten Lösungen jeweils als blaue Punkte und die Lösungen der Standardkonfiguration als rote Punkte eingezeichnet. Im linken Teil der Abbildungen sind jeweils die Lösungen aus den Experimenten in der realen Welt eingezeichnet. Die rechte Seite zeigt zum Vergleich die Lösungen aus der Simulation. Die Grafiken zeigen, dass die Lösungen beim Test auf dem realen Roboter ähnlich verteilt sind wie die Lösungen der Simulation. Die gegensätzliche Wechselbeziehung zwischen $f_{Duration}$ und $f_{SmoothTrans}$ ist auch beim realen Roboter gut zu erkennen. Außerdem ist anhand der Skalierung der Grafiken zu erkennen, dass der echte *Adept Lynx* deutlich stärker oder häufiger rotiert als der simulierte Roboter. Das kann daran liegen, dass die Unsicherheit in der realen Welt deutlich höher ist und der Fahrweg häufiger korrigiert werden muss.

Abbildung 4.23 zeigt eine erheblich höhere Streuung der Lösungen auf dem realen Roboter als im Simulator. Ferner ist in beiden Abbildungen zu erkennen, dass die Lösun-

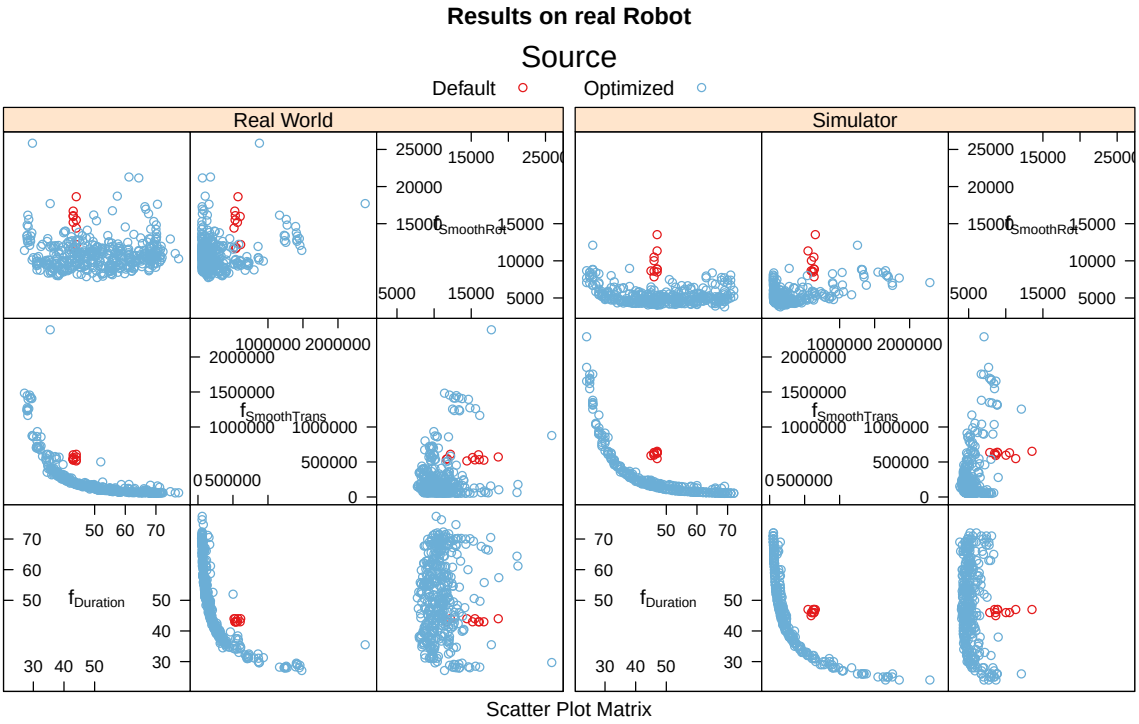


Abbildung 4.22: Vergleich der optimierten Lösungen mit der Standardkonfiguration auf dem realen Roboter und der Simulation als Scatter Plot Matrix.

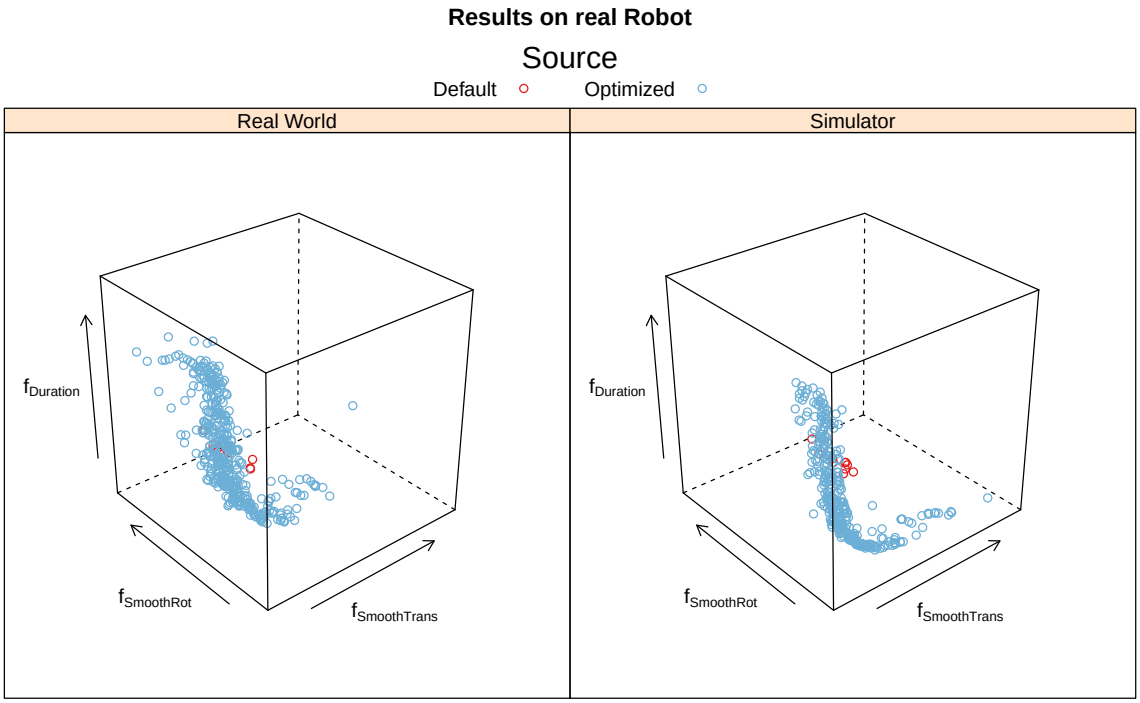


Abbildung 4.23: Vergleich der optimierten Lösungen mit der Standardkonfiguration auf dem realen Roboter und der Simulation als 3D-Punktewolke.

gen der Standardkonfigurationen in den verschiedenen Dimensionen schlechter sind als die optimierten Lösungen. Deshalb soll mittels NDS die Lösungsmenge von dominierten Lösungen bereinigt werden. Das Resultat ist eine Pareto-Front bestehend aus 150 Lösungen, die keine Lösung der Standardkonfiguration enthält. Diese 150 Lösungen sind damit echte Verbesserungen gegenüber der Standardkonfiguration.

Reproduzierbarkeit Zuletzt wird der Unterschied zwischen Simulator und *Adept Lynx* anhand der verifizierten Lösungen betrachtet. Dazu werden prozentuale Abweichungen der Zielfunktionswerte zwischen dem realen Roboter und der Simulation gebildet. Diese Abweichungen sind in Tabelle 4.11 festgehalten. Dabei gibt μ den Mittelwert, σ die Standardabweichung und $\sigma(\hat{\theta})$ den Standardfehler an.

Die Tabelle zeigt, dass die tatsächliche Fahrzeit leicht über der Fahrzeit in der Simulation liegt und recht zuverlässig optimiert werden kann. Bei den Zielfunktionen zur Optimierung der Laufruhe weichen die Werte stärker von den simulierten Ergebnissen ab. Ganz besonders stark tritt dies bei der rotatorischen Laufruhe auf. Dort verdoppeln sich die Werte. Außerdem sind sie nicht zuverlässig in der Simulation zu ermitteln, wie die hohe Standardabweichung zeigt. Dies tritt durch die erhöhte Unsicherheit der realen im Gegensatz zur simulierten Welt auf. Der Roboter muss auf diese Unsicherheiten reagieren, was bedeutet, dass er Hindernissen anders ausweicht als im Simulator. Diese Unterschiede sind von Lauf zu Lauf verschieden. Dennoch kann der Simulator zur Parameteroptimierung verwendet werden. Die Lösungen müssen allerdings nach der Optimierung auf dem *Adept Lynx* getestet werden.

Zielfunktion	μ	σ	$\sigma(\hat{\theta})$
Duration	105.07%	6.69%	0.33%
SmoothnessTrans	92.00%	17.38%	0.85%
SmoothnessRot	201.93%	46.29%	2.26%

Tabelle 4.11: Verteilung der Zielfunktionswerte auf dem realen Roboter.

Kapitel 5

Fazit und Ausblick

5.1 Fazit

Diese Arbeit demonstriert die evolutionäre Parameteroptimierung des mobilen Roboters Adept Lynx mittels eines SMS-EMOA. Die Optimierung der Roboterparameter findet in einer Simulation statt. Dadurch kann ein größerer Wertebereich der Parameter getestet werden, ohne dass der Roboter durch schlechte Parametersätze Schaden nimmt. Außerdem ist dadurch eine Parallelisierung von Auswertungen möglich. Motiviert durch die damit verbundene Zeitersparnis ist eine asynchrone Variante des SMS-EMOA entwickelt worden. Der Vorteil den SMS-EMOA asynchron zu implementieren liegt in einer Parallelisierung der Auswertungen, wobei die $(\mu + 1)$-Selektionsstrategie erhalten bleibt. Das resultiert in einer Zeitersparnis dank parallelisierter Evaluationen, während die Komplexität der Populationsaktualisierung gleich bleibt. Der entwickelte Algorithmus wurde in einem Benchmarking zum einen mit einer simultanen Ausführung mehrerer SMS-EMOAs und zum anderen mit einem unveränderten SMS-EMOA verglichen. Die Ergebnisse zeigen, dass der entwickelte Algorithmus die Lösungen mehrerer parallel ausgeführter SMS-EMOAs übertrifft und auf den verwendeten Testproblemen keine großen Qualitätsverluste gegenüber dem unveränderten Algorithmus hinnehmen muss. Er reduziert damit die Laufzeit bei Problemen für die eine parallele Auswertung möglich ist ohne signifikanten Qualitätsverlust.

Ein solches Problem ist hier gegeben. Die Parameter zur Pfadplanung des *Adept Lynx* wurden anhand von drei konfliktären Kriterien optimiert. Die Ergebnisse zeigen eine deutliche Verbesserung gegenüber der Standardkonfiguration des *Adept Lynx*. Da die Parameter auf Basis einer Simulation optimiert wurden, sind sie auf einem echten Roboter verifiziert worden. Das Ergebnis zeigt eine Verbesserung der Fahreigenschaften des *Adept Lynx*. Allerdings zeigt es auch, dass die in der Simulation ermittelten Parameter nicht ohne vorherigen Test auf den realen Roboter übernommen werden sollten.

5.2 Ausblick

Die Optimierung der Parameter kann weiter verbessert werden. Es können weitere Zielfunktionen oder Parameter zur Optimierung hinzugefügt werden. Außerdem wurden die Parameter in dieser Arbeit für eine spezielle Route optimiert. Für viele Anwendungsfälle reicht das nicht aus. Stattdessen muss eine Optimierung bezüglich der Karte geschehen. Das bedeutet, dass für eine ausreichende Optimierung der Karte eine größere Anzahl von Routen abgefahren werden muss, welche die gesamte abgebildete Umgebung ausreichend abdecken. Eventuell können diese von einem zentralen Punkt der Karte randomisiert angefahren werden, um verschiedene Strecken zu simulieren. Für diese Fälle müssen neue Testszenarien entwickelt werden. Es sollte ebenfalls untersucht werden, in wieweit zuvor händisch eingestellte Parameter durch das in dieser Arbeit vorgestellte Verfahren automatisiert weiter verbessert werden können.

Im Bezug auf den hier entwickelten Algorithmus können Studien auf weiteren Testproblemen zur Analyse seiner Leistung durchgeführt werden. Der Algorithmus könnte in diesen Studien mit einem unveränderten SMS-EMOA verglichen werden, wobei nicht die Anzahl von Auswertungen, sondern die Laufzeit begrenzt wird. Dabei kann überprüft werden, in wieweit sich die Lösungen unterscheiden und ob der asynchrone SMS-EMOA bessere Lösungen findet. Außerdem können weitere Metriken untersucht werden, welche die Leistung weiter steigern. Zurzeit werden die parallel laufenden Threads beim Zugriff auf die Population synchronisiert. Das geschieht bei der Erzeugung eines Nachkommen und bei der Aktualisierung der Population mit dem bewerteten Individuum. Es wäre denkbar, dass jeder Thread eine aktuelle Kopie der Population hält und zuerst überprüft, ob das bewertete Individuum in die lokale Population eingefügt würde. Falls das der Fall ist, wird es in die globale Population eingefügt, falls nicht, wird es direkt verworfen. Dadurch würden Threads, die versuchen gute Individuen in die globale Population einzufügen, nicht von anderen Threads aufgehalten, deren Individuen nicht in die lokale Population übernommen werden. Die Idee dabei ist, dass die lokale Population als eine leicht veraltete Kopie der globalen Population betrachtet werden kann. Daraus ergibt sich, dass ein Nachkomme, der keine Verbesserung der lokalen Population bewirkt, auch keine Verbesserung der globalen Population bewirken kann.

Anhang A

Weitere Informationen

A.1 Aufbau der Nachrichten zwischen SMS-EMOA und C#-Server-Interface

Setup-Nachricht C#-Server-Interface → SMS-EMOA
begin_Setup
Strategieparameter
begin_Individual
Entscheidungsvariablen (inklusive Grenzen)
Zielfunktionen
end_Individual
end_Setup

Evaluation-Nachricht C#-Server-Interface ↔ SMS-EMOA
begin_Evaluation
begin_Individual
Entscheidungsvariablen
Zielfunktionen
end_Individual
end_Evaluation

Ergebnis-Nachricht SMS-EMOA → C#-Server-Interface
begin_SolutionSet
Liste von Individuen
end_SolutionSet

A.2 Daten der Experimente

Algorithmus	Indikator					Zeit
	I_{HV}	I_ϵ	I_Δ	I_{GD}	I_{IGD}	
Asynchronous	4.806990	0.010177	0.688890	0.000203	0.000283	6001
Parallel	4.192673	0.297809	0.609377	0.023984	0.003986	6049
Asynchronous	4.441482	0.317394	0.698767	0.000156	0.002555	5976
Parallel	4.328851	0.209230	0.769091	0.024624	0.003614	6042
Asynchronous	4.807430	0.010782	0.667655	0.000164	0.000264	5990
Parallel	4.455278	0.181818	0.605370	0.022895	0.002515	6072
Asynchronous	4.808746	0.008868	0.695291	0.000143	0.000287	6011
Parallel	4.064168	0.381233	0.758603	0.055034	0.005651	6007
Asynchronous	4.442199	0.316984	0.694769	0.000157	0.002555	6014
Parallel	4.456560	0.193242	0.703709	0.037789	0.002568	6020
Asynchronous	4.806180	0.008963	0.689102	0.000186	0.000286	6009
Parallel	4.493078	0.177188	0.696588	0.018148	0.002218	6024
Asynchronous	4.806895	0.011211	0.679528	0.000139	0.000279	6009
Parallel	3.702031	0.721842	0.697741	0.031878	0.005496	6026
Asynchronous	4.806306	0.009737	0.678092	0.000192	0.000281	5990
Parallel	4.119893	0.324395	0.794143	0.031292	0.004506	6025
Asynchronous	4.807147	0.008875	0.676524	0.000138	0.000281	6015
Parallel	4.204085	0.283121	0.671247	0.038449	0.003893	6026
Asynchronous	4.808151	0.010914	0.669043	0.000156	0.000268	6017
Parallel	4.041580	0.335318	0.822792	0.039148	0.005596	6026

Tabelle A.1: Indikatorwerte der zehn Durchläufe der asynchronen und parallelisierten Variante des SMS-EMOA auf dem Testproblem ZDT1.

Algorithmus	**Indikator**					**Zeit**
	I_{HV}	I_ϵ	I_Δ	I_{GD}	I_{IGD}	
Asynchronous	4.806990	0.010177	0.688890	0.000203	0.000283	6001
Parallel	4.192673	0.297809	0.609377	0.023984	0.003986	6049
Asynchronous	4.441482	0.317394	0.698767	0.000156	0.002555	5976
Parallel	4.328851	0.209230	0.769091	0.024624	0.003614	6042
Asynchronous	4.807430	0.010782	0.667655	0.000164	0.000264	5990
Parallel	4.455278	0.181818	0.605370	0.022895	0.002515	6072
Asynchronous	4.808746	0.008868	0.695291	0.000143	0.000287	6011
Parallel	4.064168	0.381233	0.758603	0.055034	0.005651	6007
Asynchronous	4.442199	0.316984	0.694769	0.000157	0.002555	6014
Parallel	4.456560	0.193242	0.703709	0.037789	0.002568	6020
Asynchronous	4.806180	0.008963	0.689102	0.000186	0.000286	6009
Parallel	4.493078	0.177188	0.696588	0.018148	0.002218	6024
Asynchronous	4.806895	0.011211	0.679528	0.000139	0.000279	6009
Parallel	3.702031	0.721842	0.697741	0.031878	0.005496	6026
Asynchronous	4.806306	0.009737	0.678092	0.000192	0.000281	5990
Parallel	4.119893	0.324395	0.794143	0.031292	0.004506	6025
Asynchronous	4.807147	0.008875	0.676524	0.000138	0.000281	6015
Parallel	4.204085	0.283121	0.671247	0.038449	0.003893	6026
Asynchronous	4.808151	0.010914	0.669043	0.000156	0.000268	6017
Parallel	4.041580	0.335318	0.822792	0.039148	0.005596	6026

Tabelle A.2: Indikatorwerte der zehn Durchläufe der asynchronen und parallelisierten Variante des SMS-EMOA auf dem Testproblem ZDT3.

Algorithmus	**Indikator**					**Zeit**
	I_{HV}	I_{ϵ}	I_{Δ}	I_{GD}	I_{IGD}	
Asynchronous	383.968332	0.055367	0.727891	0.085807	0.000789	6021
Parallel	381.483599	0.909177	1.095844	2.127921	0.018242	6033
Asynchronous	383.969362	0.049873	0.556848	0.029727	0.000771	6007
Parallel	382.857413	0.538764	0.737711	0.864975	0.009666	6048
Asynchronous	383.966454	0.086117	0.886318	0.104617	0.000931	6012
Parallel	372.998746	1.426553	0.810397	1.004742	0.031855	6042
Asynchronous	383.872774	0.237094	0.631462	0.128789	0.003029	5991
Parallel	378.110531	1.207922	0.793540	3.612593	0.026575	6034
Asynchronous	383.777609	0.272779	0.770454	0.180802	0.006143	6041
Parallel	381.630590	0.729272	0.797044	1.245490	0.015773	6052
Asynchronous	383.968569	0.056784	0.609155	0.027337	0.000802	6008
Parallel	366.650975	1.964766	0.438235	2.541908	0.038602	6081
Asynchronous	383.964690	0.069330	0.649407	0.033920	0.000902	6018
Parallel	379.736120	1.138613	0.636949	1.932322	0.023978	6035
Asynchronous	383.962137	0.112966	0.785874	0.059582	0.001119	6005
Parallel	374.862961	1.594316	0.789813	1.954189	0.031074	6036
Asynchronous	383.965029	0.077020	0.723626	0.107112	0.000920	6010
Parallel	358.737931	1.852284	0.686721	1.970883	0.045756	6013
Asynchronous	383.703767	0.355662	0.841611	0.235712	0.00683	6014
Parallel	328.254713	2.877291	0.709666	2.728433	0.085976	6035

Tabelle A.3: Indikatorwerte der zehn Durchläufe der asynchronen und parallelisierten Variante des SMS-EMOA auf dem Testproblem DTLZ1.

Algorithmus	**Indikator**					**Zeit**
	I_{HV}	I_ϵ	I_Δ	I_{GD}	I_{IGD}	
Asynchronous	17.135212	0.212504	0.992780	0.001632	0.007502	5992
Parallel	14.420852	1.436508	0.751344	0.024232	0.011396	6041
Asynchronous	17.124640	0.209780	0.943261	0.001891	0.007486	6010
Parallel	15.537822	0.632735	0.632863	0.047712	0.009330	6029
Asynchronous	17.120502	0.222853	0.982491	0.001763	0.007519	6003
Parallel	15.245655	0.611769	0.592262	0.027028	0.006487	6064
Asynchronous	11.177993	2.533135	0.883320	0.001858	0.024291	5986
Parallel	16.364170	0.288283	0.536925	0.025294	0.005781	6042
Asynchronous	14.307903	1.268244	0.744610	0.002105	0.016659	5999
Parallel	13.718006	1.446814	0.513594	0.030111	0.013090	6021
Asynchronous	17.132480	0.204514	1.018406	0.001734	0.007437	6012
Parallel	14.211928	1.004255	0.721194	0.025395	0.006491	6046
Asynchronous	17.128569	0.226609	0.880758	0.001581	0.007558	6014
Parallel	16.624025	0.220746	0.497488	0.017105	0.005925	6039
Asynchronous	17.181382	0.216352	0.983220	0.001607	0.007308	6019
Parallel	14.826475	0.683647	0.684553	0.029775	0.007547	6018
Asynchronous	11.176935	2.533251	0.974099	0.001781	0.024217	5998
Parallel	13.902883	1.586952	0.824047	0.028378	0.011801	6043
Asynchronous	14.312757	1.267296	0.910229	0.002003	0.016628	5994
Parallel	15.908244	0.552555	0.637139	0.026990	0.006539	6035

Tabelle A.4: Indikatorwerte der zehn Durchläufe der asynchronen und parallelisierten Variante des SMS-EMOA auf dem Testproblem DTLZ7.

Algorithmus	**Indikator**						**Zeit**
	I_{HV}	I_ϵ	I_Δ	I_{GD}	I_{IGD}	I_{RE}	
Asynchronous	3.652013	0.015980	0.114969	0.000203	0.000349	9363	6030
Normal	3.652874	0.015911	0.108004	0.000150	0.000342	8478	29988
Asynchronous	3.649706	0.016442	0.124951	0.000321	0.000351	n.e.	6009
Normal	3.623232	0.030506	0.262600	0.000129	0.000452	9707	30075
Asynchronous	3.652317	0.015228	0.117521	0.000176	0.000340	9294	6013
Normal	3.652679	0.016960	0.144385	0.000172	0.000355	8967	29984
Asynchronous	3.652852	0.014981	0.162343	0.000141	0.000352	8525	6006
Normal	3.650626	0.015277	0.140254	0.000219	0.000347	9693	29935
Asynchronous	3.652131	0.014955	0.119852	0.000204	0.000343	9322	6022
Normal	3.653297	0.015524	0.090157	0.000169	0.000342	8388	30014
Asynchronous	3.650873	0.014814	0.136900	0.000197	0.000344	9321	6004
Normal	3.653297	0.015524	0.090157	0.000169	0.000342	8232	29982
Asynchronous	3.651949	0.015171	0.103747	0.000175	0.000347	9367	5997
Normal	3.651344	0.015437	0.119713	0.000218	0.000344	9725	30026
Asynchronous	3.649741	0.015772	0.146774	0.000204	0.000344	9388	6002
Normal	3.652747	0.014986	0.115868	0.000173	0.000344	9318	29938
Asynchronous	3.652528	0.014782	0.113397	0.000167	0.000344	8831	5993
Normal	3.652351	0.015383	0.123900	0.000162	0.000347	9105	30014
Asynchronous	3.652109	0.017334	0.130202	0.000225	0.000352	9667	6013
Normal	3.652713	0.014631	0.134423	0.000162	0.000338	8818	29984

Tabelle A.5: Indikatorwerte der zehn Durchläufe der asynchronen und der Standard-Variante des SMS-EMOA auf dem Testproblem ZDT1. Steht in der Spalte I_{RE} „n.e.“ bedeutet dies, dass die vom Indikator geforderten 98% des dominierten Hypervolumens der Pareto-Front in diesem Durchlauf nicht erreicht wurden.

Algorithmus	Indikator						Zeit
	I_{HV}	I_ϵ	I_Δ	I_{GD}	I_{IGD}	I_{RE}	
Asynchronous	4.808315	0.009373	0.683720	0.000319	0.000294	6829	6001
Normal	4.808348	0.009230	0.678459	0.000126	0.000279	6520	30011
Asynchronous	4.807740	0.012089	0.678151	0.000169	0.000272	6959	6014
Normal	4.806754	0.011356	0.671082	0.000165	0.000268	7457	29927
Asynchronous	4.441745	0.317347	0.700684	0.000139	0.002561	7622	5988
Normal	4.802606	0.012695	0.682082	0.000249	0.000270	8768	30097
Asynchronous	4.443301	0.316295	0.700936	0.000151	0.002547	7303	5993
Normal	4.746894	0.181662	0.725864	0.004900	0.002275	n.e.	30132
Asynchronous	4.806875	0.009721	0.685120	0.000168	0.000282	7777	5991
Normal	4.441315	0.317152	0.693642	0.000135	0.002542	8158	30124
Asynchronous	4.807685	0.011602	0.681422	0.000160	0.000281	7356	6004
Normal	4.807049	0.011868	0.679046	0.000156	0.000276	7105	29963
Asynchronous	4.806364	0.009103	0.664329	0.000190	0.000264	7545	5983
Normal	4.809309	0.010007	0.672623	0.000142	0.000269	6788	30073
Asynchronous	4.808106	0.010146	0.673783	0.000154	0.000273	7137	6003
Normal	4.806323	0.010474	0.676821	0.000213	0.000283	7226	29929
Asynchronous	4.807263	0.009675	0.668237	0.000164	0.000266	7204	6018
Normal	4.807374	0.010754	0.667565	0.000279	0.000283	9411	30073
Asynchronous	4.806123	0.010197	0.677963	0.000212	0.000271	7824	5995
Normal	4.808521	0.008265	0.683853	0.000175	0.000289	6615	30092

Tabelle A.6: Indikatorwerte der zehn Durchläufe der asynchronen und der Standard-Variante des SMS-EMOA auf dem Testproblem ZDT3. Steht in der Spalte I_{RE} „n.e.“ bedeutet dies, dass die vom Indikator geforderten 98% des dominierten Hypervolumens der Pareto-Front in diesem Durchlauf nicht erreicht wurden.

Algorithmus	**Indikator**						**Zeit**
	I_{HV}	I_{ϵ}	I_{Δ}	I_{GD}	I_{IGD}	I_{RE}	
Asynchronous	383.965831	0.062364	0.713325	0.067679	0.000872	n.e.	5993
Normal	383.965145	0.063866	0.697560	0.052130	0.000848	n.e.	29987
Asynchronous	383.968909	0.053769	0.452165	0.002382	0.000773	n.e.	6006
Normal	383.967031	0.055848	0.701452	0.098021	0.000832	n.e.	30072
Asynchronous	383.745725	0.267196	0.581328	0.132410	0.006233	n.e.	6011
Normal	383.963396	0.074073	0.784361	0.093439	0.000913	n.e.	30075
Asynchronous	383.751916	0.265402	0.635911	0.180364	0.006151	n.e.	6011
Normal	383.968558	0.057609	0.614995	0.080626	0.000806	n.e.	30088
Asynchronous	383.963556	0.065876	0.590649	0.041132	0.000924	n.e.	6025
Normal	383.969439	0.051069	0.597117	0.025638	0.000730	n.e.	30084
Asynchronous	383.968377	0.057363	0.505806	0.003328	0.000786	n.e.	6005
Normal	383.773288	0.272206	0.574806	0.120167	0.005530	n.e.	30025
Asynchronous	383.965482	0.065879	0.632314	0.049977	0.000953	n.e.	6006
Normal	383.967309	0.057576	0.606409	0.046252	0.000835	n.e.	30050
Asynchronous	383.751890	0.259927	0.480492	0.130459	0.006187	n.e.	5988
Normal	383.756104	0.357217	0.752611	0.283150	0.005312	n.e.	30072
Asynchronous	383.969034	0.049756	0.492826	0.011332	0.000746	n.e.	6025
Normal	383.475821	0.577498	0.867913	0.389336	0.009163	n.e.	30111
Asynchronous	383.177962	0.466005	0.527800	0.235414	0.012035	n.e.	6020
Normal	383.177768	0.468264	0.497439	0.246738	0.012025	n.e.	30183

Tabelle A.7: Indikatorwerte der zehn Durchläufe der asynchronen und der Standard-Variante des SMS-EMOA auf dem Testproblem DTLZ1. Steht in der Spalte I_{RE} „n.e.“ bedeutet dies, dass die vom Indikator geforderten 98% des dominierten Hypervolumens der Pareto-Front in diesem Durchlauf nicht erreicht wurden.

Algorithmus	**Indikator**						**Zeit**
	I_{HV}	I_ϵ	I_Δ	I_{GD}	I_{IGD}	I_{RE}	
Asynchronous	17.127760	0.212627	0.940721	0.001636	0.007543	n.e.	6015
Normal	17.123095	0.207375	0.933545	0.001609	0.007454	n.e.	30021
Asynchronous	17.135116	0.205950	0.964701	0.001523	0.007472	n.e.	6005
Normal	14.310992	1.267063	0.964773	0.001946	0.016628	n.e.	30065
Asynchronous	11.175068	2.533602	0.949167	0.001965	0.024254	n.e.	6015
Normal	17.136265	0.219780	1.024049	0.001845	0.007542	n.e.	30081
Asynchronous	17.134398	0.207219	1.001709	0.001779	0.007496	n.e.	6003
Normal	17.130509	0.208030	0.951099	0.001575	0.007473	n.e.	30025
Asynchronous	17.140645	0.208399	0.954820	0.001916	0.007480	n.e.	6006
Normal	14.313841	1.267117	0.960338	0.002062	0.016673	n.e.	30006
Asynchronous	17.140800	0.210918	0.877309	0.002084	0.007443	n.e.	5991
Normal	14.316216	1.267064	0.861442	0.002132	0.016684	n.e.	30032
Asynchronous	14.310258	1.268046	1.000059	0.001981	0.016663	n.e.	6005
Normal	14.314326	1.267332	0.958024	0.001958	0.016663	n.e.	30086
Asynchronous	17.187514	0.196112	0.984373	0.001739	0.007300	n.e.	6006
Normal	14.312448	1.267322	0.899115	0.001964	0.016623	n.e.	30000
Asynchronous	17.140284	0.205365	0.916010	0.001816	0.007471	n.e.	6005
Normal	17.141710	0.210925	0.992721	0.001942	0.007462	n.e.	30116
Asynchronous	17.129171	0.201174	0.897517	0.001588	0.007472	n.e.	6014
Normal	14.309268	1.267771	0.858646	0.002002	0.016613	n.e.	29971

Tabelle A.8: Indikatorwerte der zehn Durchläufe der asynchronen und der Standard-Variante des SMS-EMOA auf dem Testproblem DTLZ7. Steht in der Spalte I_{RE} „n.e.“ bedeutet dies, dass die vom Indikator geforderten 98% des dominierten Hypervolumens der Pareto-Front in diesem Durchlauf nicht erreicht wurden.

AIV Autonomous Indoor Vehicle

ARCL Advanced Robotics Command Language

aSMS-EMOA-Lösung Lösung der asynchronen SMS-EMOA-Variante

EA Evolutionärer Algorithmus

EMOA Evolutionärer Mehrkriterieller Optimier-Algorithmus

FTS Fahrerloses Transportsystem

GUI Graphical User Interface

NDS Non-Dominated Sorting

NSGA Nondominated Sorting Genetic Algorithm

NSGAII Nondominated Sorting Genetic Algorithm II

pSMS-EMOA-Lösung Lösung der parallelen SMS-EMOAs

uSMS-EMOA-Lösung Lösung des unveränderten SMS-EMOA

SBX Simulated Binary Crossover

SLAM Simultaneous Localization And Mapping

SMS-EMOA S-Metrik-Selektion-EMOA

SPEA2 Strength Pareto Evolutionary Algorithm 2

VEGA Vector Evaluated Genetic Algorithm

VM Virtuelle Maschine

Abbildungsverzeichnis

Algorithmenverzeichnis

Literaturverzeichnis

[1] Banwell, Paddy: *Adept Lynx Platform User's Guide.* Adept Technology, Inc., 5960 Inglewood Drive - Pleasanton, CA 94588 - USA, 11970-000 Rev D Auflage, Januar 2015.

[2] Beume, Nicola: *Hypervolumen-basierte Selektion in einem evolutionären Algorithmus zur Mehrzieloptimierung.* Diplomarbeit, Technische Universität Dortmund, Günter Rudolph–Algorithm Engineering (Ls11), März 2006.

[3] Beume, Nicola, Boris Naujoks und Günter Rudolph: *SMS-EMOA - Effektive evolutionäre Mehrzieloptimierung (SMS-EMOA - Effective Evolutionary Multiobjective Optimization).* Automatisierungstechnik, 56(7):357–364, 2008.

[4] Beume, Nicola und Günter Rudolph: *Faster S-Metric Calculation by Considering Dominated Hypervolume as Klee's Measure Problem.* In: Kovalerchuk, Boris (Herausgeber): *Proceedings of the Second IASTED Conference on Computational Intelligence*, Seiten 231–236. IASTED/ACTA Press, 2006.

[5] Coello, C.C., G.B. Lamont und D.A. van Veldhuizen: *Evolutionary Algorithms for Solving Multi-Objective Problems.* Genetic and Evolutionary Computation. Springer US, 2007.

[6] Deb, Debayan und Kalyanmoy Deb: *Investigation of Mutation Schemes in Real-Parameter Genetic Algorithms.* In: Panigrahi, BijayaKetan, Swagatam Das, PonnuthuraiNagaratnam Suganthan und PradiptaKumar Nanda (Herausgeber): *Swarm, Evolutionary, and Memetic Computing*, Band 7677 der Reihe *Lecture Notes in Computer Science*, Seiten 1–8. Springer Berlin Heidelberg, 2012.

[7] Deb, Kalyanmoy: *Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems.* Evolutionary Computation, 7(3):205–230, 1999.

[8] Deb, Kalyanmoy: *Multi-Objective Optimization using Evolutionary Algorithms.* Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons, Chichester, 2001.

[9] Deb, Kalyanmoy und Hans-georg Beyer: *Self-Adaptive Genetic Algorithms with Simulated Binary Crossover.* Evolutionary Computation, 9(2):197–221, Juni 2001.

[10] Deb, Kalyanmoy und Amarendra Kumar: *Real-coded Genetic Algorithms with Simulated Binary Crossover: Studies on Multimodal and Multiobjective Problems.* In: Rowland, Todd (Herausgeber): *Complex Systems*, Band 9, Seiten 431–454. Complex Systems Publ., 1995.

[11] Deb, Kalyanmoy, Amrit Pratap, Sameer Agarwal und T. Meyarivan: *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II.* IEEE Transactions on Evolutionary Computation, 6:182–197, 2002.

[12] Deb, Kalyanmoy, Lothar Thiele, Marco Laumanns und Eckart Zitzler: *Scalable Test Problems for Evolutionary Multi-Objective Optimization.* Technischer Bericht 112, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH, Gloriastrasse 35., ETH-Zentrum, CH-8092, Zürich, Switzerland, Juli 2001.

[13] Durillo, Juan J. und Antonio J. Nebro: *jMetal: A Java framework for multi-objective optimization.* Advances in Engineering Software, 42:760–771, 2011.

[14] Emmerich, Michael, Nicola Beume und Boris Naujoks: *An EMO Algorithm Using the Hypervolume Measure as Selection Criterion.* In: Coello, Carlos A. Coello, Arturo Hernández Aguirre und Eckart Zitzler (Herausgeber): *Evolutionary Multi-Criterion Optimization, Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005, Proceedings*, Band 3410 der Reihe *Lecture Notes in Computer Science*, Seiten 62–76. Springer, 2005.

[15] Fogel, L.J., A.J. Owens und M.J. Walsh: *Artificial intelligence through simulated evolution.* Wiley, Chichester, WS, UK, 1966.

[16] Fox, Dieter, Wolfram Burgard und Sebastian Thrun: *The dynamic window approach to collision avoidance.* IEEE Robot. Automat. Mag., 4(1):23–33, 1997.

[17] Goldberg, David E.: *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co., Inc., New York, NY, USA, 1989.

[18] Holland, J. H.: *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, USA, 1975.

[19] Kolda, TamaraG. und VirginiaJ. Torczon: *Understanding Asynchronous Parallel Pattern Search.* In: Di Pillo, Gianni und Almerico Murli (Herausgeber): *High Performance Algorithms and Software for Nonlinear Optimization*, Band 82 der Reihe *Applied Optimization*, Seiten 323–342. Springer US, 2003.

[20] Koza, John R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, USA, 1992.

[21] Papageorgiou, Markos, Marion Leibold und Martin Buss: *Allgemeine Problemstellung der statischen Optimierung.* In: *Optimierung*, Seiten 11–17, 2012.

[22] Rechenberg, I.: *Evolutionsstrategie 94*, Band 1 der Reihe *Werkstatt Bionik und Evolutionstechnik.* Frommann-Holzboog, Stuttgart, 1994.

[23] Schaffer, J. David: *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms.* In: *Proceedings of the 1st International Conference on Genetic Algorithms*, Seiten 93–100, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc.

[24] Thrun, Sebastian und John J. Leonard: *Simultaneous Localization and Mapping.* In: Siciliano, Bruno und Oussama Khatib (Herausgeber): *Springer Handbook of Robotics*, Seiten 871–889. Springer Berlin Heidelberg, 2008.

[25] Vaughan, Richard: *Massively multi-robot simulation in stage.* Swarm Intelligence, 2(2-4):189–208, 2008.

[26] Wessing, Simon: *Towards Optimal Parameterizations of the S-Metric Selection Evolutionary Multi-Objective Algorithm.* Diplomarbeit, Technische Universität Dortmund, Günter Rudolph–Algorithm Engineering (Ls11), Juli 2009.

[27] Zhang, Xingyi, Ye Tian, Ran Cheng und Yaochu Jin: *An Efficient Approach to Nondominated Sorting for Evolutionary Multiobjective Optimization.* IEEE Trans. Evolutionary Computation, 19(2):201–213, 2015.

[28] Zitzler, E., K. Deb und L. Thiele: *Comparison of Multiobjective Evolutionary Algorithms: Empirical Results.* Evolutionary Computation, 8(2):173–195, 2000.

[29] Zitzler, Eckart, Marco Laumanns und Lothar Thiele: *SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization.* In: *Evolutionary Methods for Design, Optimisation, and Control*, Seiten 95–100. CIMNE, Barcelona, Spain, 2002.

[30] Zitzler, Eckart und Lothar Thiele: *Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study.* In: *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, PPSN V, Seiten 292–304, London, UK, 1998. Springer-Verlag.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den 7. Dezember 2015

Dino Alexander Menges