Masterarbeit

Entwurf interplanetarer Bahnkurven mittels Hint-basierter Optimierung

Oskar Pusz

Hannover, den 06.07.2016

Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Systems Engineering
Fachgebiet System- und Rechnerarchitektur

Erstprüfer:	Prof. Dr.-Ing. Christian Müller-Schloer
Zweitprüfer:	Prof. Dr.-Ing. Bernardo Wagner
Betreuer:	Dipl.-Math. Sebastian Niemann

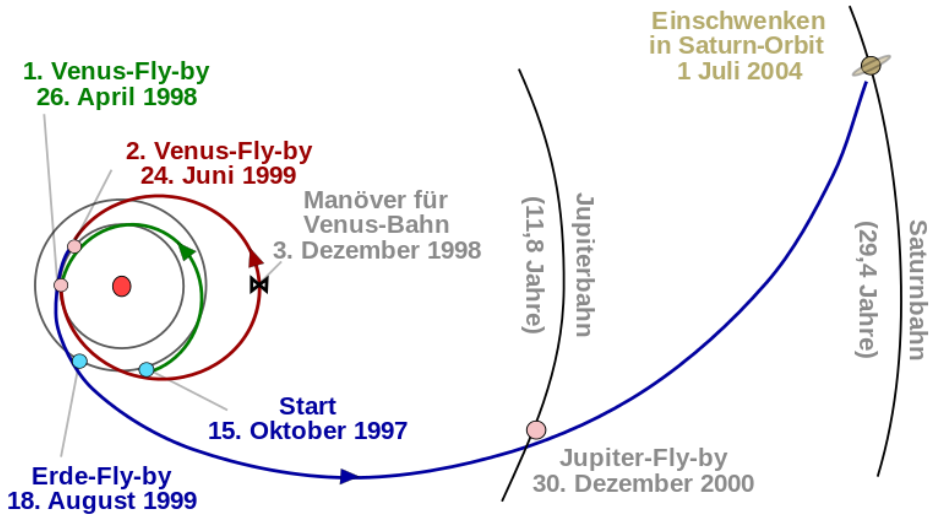
Gottfried Wilhelm Leibniz Universität Hannover,
Institut für Systems Engineering – System- und Rechnerarchitektur
Appelstr. 4, 30167 Hannover

Masterarbeit

Entwurf interplanetarer Bahnkurven mittels Hint-basierter Optimierung

Der Entwurf interplanetaren Bahnkurven behandelt die Planung mehrerer Bahnmanöver (z.B. Gravitationsmanöver) eines interplanetaren Flugkörpers im Weltraum zwischen zwei Himmelskörpers.

Aufgrund der stark begrenzten Menge an verfügbaren Treibstoff und zur Minimierung der Missionskosten sind interplanetare Bahnkurven ein genereller Gegenstand der Optimierung.



Flugbahn der Cassini-Huygens Raumsonden von der Erde bis zum Saturn.

In dieser Arbeit soll die von der Europäischen Weltraumbehörde (ESA) proklamierte Mission *Save the Earth* der ersten Global *Trajectory Optimisation Competition* (GTOC) behandelt werden.

In dieser Mission soll ein Kollisionsmanöver eines Satelliten mit dem Asteroiden *2001 TW229* unter Maximierung der Aufschlagsenergie geplant werden. Der Wettbewerb basiert in Teilen auf der von der ESA und der NASA gemeinsam für 2020 geplanten *Asteroid Impact and Deflection Assessment* (AIDA) Mission.

Das Ziel der Arbeit umfasst somit:

- Implementierung des GTOC 1 Problems auf Basis der Vorgaben der ESA.
- Untersuchung der Eignung Hint-basierter Optimierung für dieses Problem.

Die Bearbeitung der Abschlussarbeit soll in C++11/14 auf Basis der am SRA entwickelten Optimierungs-Bibliothek Mantella erfolgen.

Fakultät für Elektrotechnik und Informatik

Fachgebiet System- und Rechnerarchitektur (SRA)

Sebastian Niemann

Tel. +49 511 762 19732
niemann@sra.uni-hannover.de

Besucheradresse:
Appelstr. 4
30167 Hannover
www.sra.uni-hannover.de

Zentrale:
Tel.: +49 511 762

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig und ausschließlich mit Hilfe der angegebenen Quellen und Hilfsmittel angefertigt habe.

Hannover, den 06.07.2016 Oskar Pusz

Zusammenfassung

In dieser Masterarbeit wird untersucht, ob sich das Optimierungsverfahren *Hint-Based Optimisation* für die automatisierte Auswahl eines passenden Algorithmus bezüglich der Qualität der gefundenen Lösung eines Optimierungsproblems lohnt. Hierzu wird eine Problemfamilie des Wettbewerbs *Global Trajectory Optimisation Competition* ausgewählt, welche die Berechnung interplanetarer Bahnkurven behandelt. Lösungen innerhalb dieser Problemfamilie werden von sechs Algorithmen gesucht, die von *Hint-Based Optimisation* ausgewählt werden sollen. Ziel der Arbeit ist im Rahmen dieser Problemfamilie zu analysieren, ob *Hint-Based Optimisation* in der Lage ist solch eine Wahl passend zu treffen oder ob ein weniger komplexes Optimierungsverfahren besser damit umgehen kann. Dazu werden im Endeffekt aus experimentellen Messungen Rankings der Algorithmen aufgestellt, die mit den vorgeschlagenen Rankings beider Optimierungsverfahren abgeglichen werden. Dadurch zeigt sich, dass beide Verfahren die Rankings gut vorhersagen können, wobei *Hint-Based Optimisation* im Vergleich besser abschneidet.

Abstract

This master thesis will expose whether the optimisation method *Hint-Based Optimisation* is eligibly for an automated choice of an appropriate algorithm in respect to the quality of the found solution of an optimisation problem. A problem family of the *Global Trajecctory Optimisation Competition* will be selected which deal with the domain of calculating interplanetar trajectories. Six algorithms will search in this problem family for solutions which should be choosen by *Hint-Based Optimisation*. The target of this thesis is to analyse whether *Hint-Based Optimisation* is able to make a right choice for an algorithm or a less complex method will deal better with this. After that, ranks of measured values will be compared to the suggested ones of both optimisation methods. This makes it clear that both optimisation methods predict rankings well, but *Hint-Based Optimisation* gain the advantage in comparison.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

$\mathcal{D}$	Definitionsbereich
$\mathcal{W}$	Wertebereich
$\mathcal{L}$	Lösungsraum
$\mathcal{O}$	Menge optimaler Lösungen
$U(a, b)$	Uniforme verteilte Zufallsvariable aus $[a, b)$
$\mathbf{r}$	Positionsvektor
$\mathbf{v}$	Geschwindigkeitsvektoren
n	Problemdimension
$\mathbf{s}$	Flugsequenz-Vektor
l	Flugsequenzlänge
t_0	Abflugszeit
t_i	Transferzeit i
$\mathbf{t}$	Zeitvektor, beinhaltet Abflugs- und Transferzeiten
F	(Aufschlags-)Kraft
$f, f(x)$	Zielfunktion
$\tilde{f}$	Akzeptanzwert bezüglich der Funktion f
$f_s, f_s(x)$	Konkrete Zielfunktion durch Flugsequenz $\mathbf{s}$
$\mathbf{A}$	Menge aus Gütetupel von Algorithmen
$\mathbf{a}_i$	Gütetupel eines Algorithmus i
b	Anzahl der Elemente in $\mathbf{A}$
$\mathbf{p}$	Eigenschaftsvektor aus Hint-based Optimisation
p_i	Eigenschaft i des Eigenschaftsvektors
m	Länge von $\mathbf{p}$

Legende

x, x_i	Skalar	X	Matrix
$\mathbf{x}, \mathbf{x}_i$	Vektor	$\mathbf{X}$	Menge

Abkürzungsverzeichnis

AM	Adaptives Mapping
ASP	Algorithm Selection Problem
DE	Differential Evolution
DL	Direct Learning
ESA	European Space Agency
GTOC	Global Trajectory Optimisation Competition
HC	Hill Climber
HINT	Hint-based Optimisation
HJ	Hooke-Jeeves-Algorithmus
PCA	Patched Conic Approximation
PSO	Particle Swarm Optimization
SA	Simulated Annealing

1 Einführung

1.1 Motivation

In der Wissenschaft und Wirtschaft entstehen mathematische Probleme, die oft komplex und nicht von Hand lösbar sind. So wird auf Optimierungsalgorithmen zurückgegriffen, die es ermöglichen nach *optimalen* Lösungen für das Problem zu suchen. Schon seit 1995 sind mindestens 49 Optimierungsalgorithmen veröffentlicht worden und dabei kommt oft die Frage auf welcher bezüglich der Laufzeit oder Qualität der Lösung geeignet ist [1, 2]. Diese generelle Frage wird auch *Algorithm Selection Problem* (ASP) genannt, für das es keine Pauschallösung gibt. Damit gekoppelt ist das *No Free Lunch Theorem*, dass im Kern besagt, dass über alle denkbaren Optimierungsprobleme betrachtet jeder Algorithmus im Mittel die gleiche Anzahl an Iterationen bei fester Qualität der zu findenden Lösung benötigt und im Umkehrschluss somit kein Algorithmus im Mittel über alle Probleme besser als die Zufallssuche oder das Raten der immer gleichen Lösung ist [3, 4]. Es existieren aber dennoch Teilklassen von Optimierungsproblemen für die sich geeignete Algorithmen finden lassen [5, 6].

Die Wahl eines passenden Algorithmus wird oft auf aufwändige Analysen des Problems gestützt und kann sehr zeit- und/oder kostenintensiv sein [7]. Um darauf verzichten zu können, ist eine automatisierte Methode gefragt, die in der Lage ist den *richtigen* Algorithmus ohne Vorwissen oder vorherige Analysen auszuwählen. Ein mögliches Anwendungsgebiet ist die Bestimmung interplanetarer Bahnkurven, das in dieser Arbeit näher behandelt wird. Missionen im Weltall sind sehr kostenintensiv, sodass bei der Berechnung solcher Bahnkurven die Qualität der Lösung im Vordergrund liegt, damit die dazugehörige Mission erfolgreich wird. Dazu veröffentlicht die *European Space Agency* selbst bezeichnete *nearly-impossible optimisation problems*, die zur Berechnung interplanetarer Bahnkurven führen [8]. Auch in diesem Anwendungsgebiet werden weiterhin für die verschiedenen Arten von Problemen Algorithmen gesucht, die das jeweilige Problem effizient lösen [6].

1.2 Ziel der Arbeit

In dieser Arbeit wird der Ansatz *Hint-Based Optimisation* (HINT) betrachtet, der versucht das ASP automatisiert zu *lösen*. Bis dato wurde dieser Ansatz für Probleme aus der Robotik untersucht. In dieser Betrachtung von HINT standen Lösungen des Problems im Fokus, die mit akzeptablem Fehler in minimaler Zeit gefunden werden sollen. Dem entsprechend wurde dort

das ASP mit HINT bezüglich dieser Art von Problemen untersucht. Dadurch ist es möglich mit der Zeit einen Algorithmus zu finden, der die geforderten Kriterien der minimalen Anzahl an Iterationen und der Mindestqualität gefundener Lösungen erfüllt.

Das Ziel dieser Arbeit ist eine gänzlich andere Familie an Problemen zu untersuchen. Die Problemfamilie behandelt Berechnungen von interplanetaren Bahnkurven, das heißt Flugbahnen im Weltall von beispielsweise Raketen oder Raumschiffen. Dabei steht hier nicht wie bei den Problemen der Robotik die minimale Laufzeit von Optimierungsalgorithmen im Fokus, sondern die Maximierung des Funktionswertes der gefundenen Lösungen. Missionen im Weltall sind sehr teuer, sodass bei der Planung von Flugbahnen die Qualität der Lösung im Mittelpunkt und die Laufzeit der Optimierungsalgorithmen im Hintergrund steht. Auch hier soll mit Hilfe von HINT das ASP näher untersucht werden. Dazu wird der hier zu untersuchende Ansatz mit einem simpleren Verfahren verglichen um zu prüfen, ob sich der Aufwand mit HINT gegenüber dem simpleren Verfahren gelohnt hat, bzw. ob einer dieser beiden Ansätze überhaupt geeignet ist.

1.3 Struktur der Arbeit

Zu Beginn wird in Kapitel 2 auf einige Grundlagen eingegangen um ein Grundverständnis zu schaffen. Darin wird erläutert was Optimierung ist und wie die in dieser Arbeit verwendeten Optimierungsalgorithmen arbeiten. Ebenfalls wird die Domäne der zu untersuchenden Problemfamilie vorgestellt. Die Erläuterung beinhaltet Mechaniken und dazugehörigen mathematischen Probleme, auf die in den Grundlagen näher eingegangen wird. Anschließend wird in Kapitel 3 auf die sogenannte *Global Trajectory Optimisation Competition* (GTOC) eingegangen. Es wird angesprochen was diesen Wettbewerb ausmacht und welches Problem in dieser Arbeit näher betrachtet wird. Nach den Grundlagen und Beschreibung des Wettbewerbs folgt in Kapitel 4 der Stand der Technik. Hier wird beschrieben auf welchen aktuellen Stand die Forschung bzgl. des ASP und der Domäne interplanetarer Bahnkurven. Fokus dieser Arbeit ist das ASP. Dazu werden Optimierungsverfahren eingeführt, die im Laufe der Zeit Optimierungsalgorithmen bestimmen sollen, die zu den Problemen aus der Domäne der interplanetaren Bahnkurven passen. Es soll dazu HINT evaluiert werden und somit wird dieser Ansatz in Kapitel 5 als Optimierungsverfahren vorgestellt. Es wird in diesem Kapitel ebenfalls eine simplere Art des Ansatzes beschrieben, die HINT gegenübergestellt wird. Anschließend folgt in Kapitel 6 die Evaluation der Optimierungsverfahren an sich. Es beginnt mit der Beschreibung der Systematik der Analyse. Daraufhin wird auf Details der Vorgehens eingegangen und abschließend werden die Ergebnisse zusammengetragen und diskutiert. Zum Schluss der Arbeit folgt in Kapitel 7 ein Fazit und in Kapitel 8 ein Ausblick für weitere Betrachtungen.

2 Grundlagen

2.1 Optimierung

In diesem Abschnitt wird als Erstes auf die Optimierung eingegangen und anschließend auf einige bekannte Optimierungsalgorithmen, die in dieser Arbeit benutzt werden.

2.1.1 Optimierungsproblem

Bei einer kontinuierlichen, reellwertigen Optimierung existiert ein Definitionsbereich $\mathcal{D} \in \mathbb{R}^n$ und ein Wertebereich $\mathcal{W} \in \mathbb{R}$. Einschränkungen durch Nebenbedingungen wie Intervallbeschränkungen a und b von $\mathcal{D}$ bestimmen einen Lösungsraum $\mathcal{L} \coloneqq \{x \in \mathcal{D} | a \leq x \leq b\}$. Die zum Optimierungsproblem gehörige Zielfunktion f ist damit eine Abbildung der Form $f : (\mathcal{L} \subseteq \mathcal{D}) \rightarrow \mathcal{W}$. Im Allgemeinen ist das Ziel der Optimierung die Lokalisierung einer optimalen Lösung in $\mathcal{L}$ der Zielfunktion f (je nach Kontext Minimum oder Maximum). Es wird ein Akzeptanzwert $\tilde{f}$ eingeführt, der definiert wann gefundene Lösungen als optimal gelten. Damit entsteht die Menge an optimalen Lösungen $\mathcal{O} \coloneqq \{x \in \mathcal{L} | f(x) \leq \tilde{f}\}$. Die Suche des Optimums in einem Suchraum hoher Dimension kann sich als sehr aufwändig gestalten, sodass effiziente Mechaniken angewendet werden müssen um in absehbarer Zeit eine optimale Lösung in $\mathcal{O}$ zu finden. Es wird zwischen lokalen Optima und globalen Optima unterschieden. An lokalen wie auch globalen Optima ist der Gradient der Nullvektor, aber ein globales Optimum ist je nach Kontext der beste Wert des Optimierungsproblems. Ein globales Optimum einer Zielfunktionen zu finden ist bei vielen Optimierungsproblemen eine Herausforderung, da sie meist eine Zielfunktion behandeln, die schwer handelbare Eigenschaften wie unregelmäßige Steigungsschwankungen bzw. eine hohe Anzahl an lokalen Optima aufweisen, was die Suche nach einem globalen Optimum sehr aufwendig bis unmöglich macht. Aus diesem Grund wird nach einem Akzeptanzwert $\tilde{f}$ optimiert. In dieser Arbeit werden sogenannte *Black-Box Probleme* betrachtet, in der eine minimierte Lösung gefunden werden soll, die den Akzeptanzwert $\tilde{f}$ unterschreitet. Bei dieser Art von Problemen ist keine Ableitung der Zielfunktion oder Information über dessen Stetigkeit verfügbar [9].

2.1.2 Optimierungsalgorithmen

Optimierungsalgorithmen haben das Ziel die zuvor genannten Optimierungsprobleme zu lösen, das heißt im Lösungsraum nach einer ausreichend optimalen Lösung zu suchen. Falls ein Maximierungsproblem mit Zielfunktion f vorliegt, kann die Zielfunktion und Akzeptanzwert $\tilde{f}$

negiert werden, was äquivalent eines Minimierungsproblems mit Zielfunktion $-f$ und $-\tilde{f}$ ist. Somit ist es nicht relevant, ob die Algorithmen nach Maxima oder Minima suchen, da das Optimierungsproblem mit einer Negation leicht umformuliert werden kann.

In dieser Arbeit werden ableitungsfreie Optimierungsalgorithmen genutzt, da hier *Black-Box Probleme* betrachtet werden. Das heißt, dass zu der Zielfunktion keine Ableitung oder Information zur Stetigkeit der Funktion benötigt werden. Die Ableitung einer Funktion oder Informationen über die Stetigkeit zu bestimmen ist in der Regel nur dann möglich, wenn die Funktion analytisch darstellbar ist.

Für alle Optimierungsalgorithmen gilt: Sie iterieren entweder solange bis ein Wert $f(\mathbf{x})$ mit $\mathbf{x} \in \mathcal{O}$ gefunden wurde oder die maximale Anzahl an Funktionsauswertungen erreicht ist. Einige Algorithmen besitzen auch Parameter, die das Verhalten des Algorithmus festlegen und einen großen sowie chaotischen Einfluss auf die Laufzeit oder Qualität der Lösung haben können.

Es werden die Zufallssuche und fünf gängige Optimierungsalgorithmen in dieser Arbeit genutzt. Bei den vorgestellten Algorithmen wird von einem zu lösenden Minimierungsproblem ausgegangen:

- Die **Zufallssuche** würfelt für jede Dimension des Wertebereichs $\mathcal{W}$ uniform verteilte Zufallswerte. Dieser Algorithmus verfügt über keinerlei Gedächtnis oder Logik. Üblicherweise wird die Zufallssuche als der schlechteste Algorithmus angesehen, sodass alle entwickelten Algorithmen immer besser als die Zufallssuche sein sollten.

- In jeder Iteration bestimmt **Hill Climbing** innerhalb einer n-dimensionalen Sphäre um $\mathbf{x}_{akt}$ einen uniform verteilten Kandidaten $\mathbf{x}_{cand}$. Ist der Funktionswert $f(\mathbf{x}_{cand})$ kleiner als $f(\mathbf{x}_{akt})$, so wird $\mathbf{x}_{akt}$ durch $\mathbf{x}_{cand}$ ersetzt. Der Radius r der n-dimensionalen Sphäre ist konfigurierbar. Der Ablauf einer Iteration ist folgender [10]:

  $\mathbf{x}_{cand} = \text{randomNeigbour}(\mathbf{x}_{akt}, r)$
  if $f(\mathbf{x}_{cand}) < f(\mathbf{x}_{akt})$ **then**
  $\quad \mathbf{x}_{akt} = \mathbf{x}_{cand}$
  end if

- **Simulated Annealing** lässt sich als *simulierte Abkühlung* übersetzen und ist ähnlich wie der Hill Climbing aufgebaut. Allerdings ist die Strategie der Schrittweite und die Wahl der neuen aktuellen Position unterschiedlich. Es existiert eine streng monoton fallende Sequenz aus *Temperaturen*, im Allgemeinen eine Schrittweite, mit der die aktuelle Position verrechnet wird und so ein Kandidat für eine neue Position entsteht, beschreibt. Neue Positionen mit einem besseren Funktionswert werden als aktuelle Postionen übernommen.

Falls die neue Postion schlechter sein sollte, besteht eine mit der aktuellen *Temperatur* t und der Differenz aus den Funktionswerten Δf gekoppelte Wahrscheinlichkeit $e^{\Delta f/t}$, dass diese Position dennoch als aktuelle übernommen wird um die Streuung der Suche breit zu halten. Es werden alle *Temperaturen* durchlaufen bis die *Temperatur* 0 erreicht ist. Die Temperatursequenz ist konfigurierbar. Der Ablauf einer Iteration i bei einer maximalen Anzahl Iterationen i_{max} ist folgender [11]:

$t = i/i_{max}$
$\mathbf{x}_{cand} = \text{randomNeighbor}(\mathbf{x}_{akt}, t)$
if $f(\mathbf{x}_{cand}) < f(\mathbf{x}_{akt})$ **then**
  $\mathbf{x}_{akt} = \mathbf{x}_{cand}$
else
  $\Delta f = f(\mathbf{x}_{akt}) - f(\mathbf{x}_{cand})$
  if $U(0,1) > e^{\Delta f/t}$ **then**
    $\mathbf{x}_{akt} = \mathbf{x}_{cand}$
  end if
end if

- Der **Hooke-Jeeves Algorithmus** (auch *Pattern Search* genannt) bestimmt in jeder einzelnen Dimension k in positiver wie auch negativer Richtung mit einer bestimmten Distanz d eine neue Postion im Lösungsraum. D.h. es werden die Kandidaten $(x_0 + d, x_1, x_2, \ldots, x_n)$, $(x_0 - d, x_1, x_2, \ldots, x_n)$, $(x_0, x_1 + d, x_2, \ldots, x_n)$, $(x_0, x_1 - d, x_2, \ldots, x_n)$ usw. auf einen besseren Funktionswert geprüft. Ist einer dieser Kandidatenpositionen besser als die aktuelle Position, so wird diese als aktuelle übernommen und es neu angefangen zu suchen. Falls keiner dieser $2n$ Kandidatenpostionen besser als die aktuelle ist, wird die Distanz d halbiert und die Prüfung der $2n$ Punkte beginnt von neuem. Die Distanz d ist konfigurierbar. Die Kandidaten sind in einem Container X gespeichert und X_k entspricht dem k-ten Kandidaten. Der Ablauf einer Iteration ist folgender [12]:

for dimension k **in** n **do**
  $\mathbf{x}_{tmp} = \mathbf{x}_{akt}$
  $x_{tmp,k} = x_{tmp,k} + d$
  $X_{2k-1} = \mathbf{x}_{tmp}$
  $x_{tmp,k} = x_{tmp,k} - 2d$
  $X_{2k} = \mathbf{x}_{tmp}$
end for
$\mathbf{x}_{best} = \mathbf{x}_{akt}$
$y_{best} = \infty$
for candidate $\mathbf{x}_{cand}$ **in** X **do**
  if $f(\mathbf{x}_{cand}) < y_{best}$ **then**
    $\mathbf{x}_{best} = \mathbf{x}_{cand}$
    $y_{best} = f(\mathbf{x}_{cand})$
  end if
end for
if $\mathbf{x}_{best} \Leftrightarrow \mathbf{x}_{akt}$ **then**
  $d = d : 2$
else
  $\mathbf{x}_{akt} = \mathbf{x}_{best}$
end if

- **Particle Swarm Optimization** ist ein sogenannter Schwarm-Algorithmus. Es gibt einen Schwarm aus einer festgelegten Anzahl von Partikeln, die sich im Lösungsraum fortbewegen. Jeder Partikel i kennt seine aktuelle $\mathbf{x}_{akt,i}$, die eigene $\mathbf{x}_{p,i}$ und global beste $\mathbf{x}_g$ Position mit dem jeweiligen Funktionswert und seine eigene Geschwindigkeit v_i. Die Geschwindigkeit beschreibt, verrechnet mit der aktuellen Position, die neue Position des Partikels im nächsten Iterationsschritt. Des Weiteren wirken die eigenen und global besten Positionen als Anziehungskräfte c_p und c_g, sodass in der Berechnung stets die globalen Informationen anderer Partikel miteinbezogen und kommuniziert werden müssen. Zusätzlich dazu gibt es noch einen Beschleunigungsfaktor λ, der bestimmt wie stark sich die berechnete Bewegung auf die aktuelle Position der Partikel auswirken soll. Die Anzahl an Partikeln, die Gewichtung der Anziehungskräfte c_p und c_g sowie der Beschleunigungsfaktor λ sind konfigurierbar. Der Ablauf einer Interation ist folgender [13]:

for particle i **in** p **do**
  $\mathbf{v}_i = \lambda \mathbf{v}_i + c_p(\mathbf{x}_{p,i} - \mathbf{x}_{akt,i}) + c_g(\mathbf{x}_g - \mathbf{x}_{akt,i})$
  $\mathbf{x}_{akt,i} = \mathbf{x}_{akt,i} + \mathbf{v}_i$
  if $f(\mathbf{x}_{akt,i}) < f(\mathbf{x}_{p,i})$ **then**
    $\mathbf{x}_{p,i} = \mathbf{x}_{akt,i}$
    if $f(\mathbf{x}_{p,i}) < f(\mathbf{x}_g)$ **then**
      $\mathbf{x}_g = \mathbf{x}_{p,i}$
    end if
  end if
end for

- **Differential Evolution** ist zu einem Schwarm-Algorithmus ähnlich, wobei sich die Strategie der Suche nach neuen Positionen unterscheidet. Es existiert eine beliebig große Population an Partikeln. Jeder Partikel i aus den insgesamt p Partikeln durchläuft pro Iteration die folgenden drei Schritte: In der *Mutation* wird die aktuelle Position mit der Positionen zufälliger Partikel r_1 und r_2 verrechnet und es entsteht der Mutationsvektor $\mathbf{m}_i$. Der Einfluss der Verrechnung ist gewichtet und heißt Mutationsfaktor c_m. Beim *Crossover* wird eine über den Crossoverfaktor c_c konfigurierbare Wahrscheinlichkeit bestimmt welche der einzelnen Dimensionen k aus dem mutierten Vektor als aktuelle Komponenten der aktuellen Position tatsächlich übernommen werden sollen. Zum Schluss folgt die *Selection*, die die Postion nach Durchlaufen der Mutation und dem Crossover prüft, ob ein besserer Funktionswert gefunden wurde. Falls ja, wird die neue Position als aktuelle übernommen, sonst bleibt es bei der ursprünglichen aktuellen Position. Die Größe der Population und die Mutation- und Crossoverfaktoren sind konfigurierbar. Der Ablauf einer Iteration ist folgender [14]:

for particle i **in** p **do**
  $\mathbf{m}_i = \mathbf{x}_i + c_m(\mathbf{x}_{r1} - \mathbf{x}_{r2})$ ▷ Mutation
  for dimension k **in** n **do** ▷ Crossover
    if $U(0,1) \geq c_c \vee i = \lfloor U(1, p+1) \rfloor$ **then**
      $x_{neu,i,k} = m_{i,k}$
    end if
  end for
  if $f(\mathbf{x}_{neu,i}) < f(\mathbf{x}_{akt,i})$ **then** ▷ Selection
    $\mathbf{x}_{akt,i} = \mathbf{x}_{neu,i}$
  end if
end for

2.2 Himmelsmechanik

Diese Arbeit beschäftigt sich mit der Optimierung von Flugbahnen im Weltall. Um bestimmte Begrifflichkeiten und physikalische/mathematische Zusammenhänge und ihre Komplexität verstehen zu können, werden hier Grundlagen zur Himmelsmechanik beschrieben. Es folgt anfangs die Beschreibung eines in der Astrophysik üblichen Kalenders. Anschließend werden Himmelskörper und ihre möglichen Flugbahnen im Allgemeinen beschrieben. Abschließend werden Probleme vorgestellt, die in der Himmelsmechanik stückweise bis zum direkten Modell der Flugbahnbestimmung abgebildet werden.

2.2.1 Julianisches Datum

Das im Alltag gebräuchliche *Gregorianische Datum* (GD) beschreibt einen Zeitpunkt samt Jahr, Monat, Tag und Uhrzeit. Im Gegensatz dazu beschreibt das julianische Datum (JD) eine in der Astrophysik gängige kontinuierliche Tageszählung. Die Zählung beginnt mit dem 1. Januar 4713 v. Chr. 12:00 Uhr GD, was entsprechend dem julianischen Datum 0 entspricht. Mit jedem vergangenen Tag wird das Datum um eins hochgezählt, wobei auch reelle Zahlenbereiche zur Datumsangabe in JD zulässig sind. Zum Beispiel entspricht der 6. Juli 2016 n. Chr. 10:15 Uhr GD dem julianische Datum 2457575.927083. Desweiteren existiert das modifizierte julianische Datum (MJD) um den Zahlenbereich allgemein kleiner zu halten. Das Datum 0 MJD wurde dabei auf den 17. November 1858 n. Chr. 00:00 Uhr GD gesetzt. Es gilt durch die Festsetzung dieser Daten folgende Gleichung: MJD $=$ JD - 2400000.5. Das modifizierte julianische Datum 2000 (MJD2000) gilt als die letzte übliche kontinuierliche Tageszählung, wobei das Datum 0 MJD2000 auf den 1. Januar 2000 n. Chr. 00:00 Uhr GD festgefest worden ist. Ein kurze abschließende tabellarische Übersicht ist in Tabelle 2.1 zu sehen.

GD	JD	MJD	MJD2000
01. Jan. 4713 v. Chr. 12:00 Uhr	0	-2400000.5	-2451544.5
17. Nov. 1858 n. Chr. 00:00 Uhr	2400000.5	0	-51544
01. Jan. 2000 n. Chr. 00:00 Uhr	2451544.5	51544	0
06. Jul. 2016 n. Chr. 10:15 Uhr	2457575.927	57575.427	6031.427

Tabelle 2.1: Übersicht verschiedener Arten julianischer Daten samt Beispiel

2.2.2 Himmels- und Flugkörper

Als direkte Beispiele für Himmelskörper dienen Sonnen, Monde, Planeten, Asteroiden, etc. In anderen Worten sind es alles astronomische Körper, die eine Masse aufweisen und sich träge auf einer Flugbahn bewegen. Flugkörper hingegen sind z.B. Satelliten, Raumschiffe und Raketen die über eine Art Rückstoßantrieb verfügen um Geschwindigkeit und Richtung von Flugbahnen während des Fluges ändern zu können.

Wie in vielen Ausarbeitungen üblich wird Position und Beschleunigung eines Himmels- oder Flugkörpers zu einem bestimmten Zeitpunkt im julianischen Kalender durch einen Positionsvektor **r** und einen Geschwindigkeitsvektor **v** definiert. Die zeitliche Relation des Geschwindigkeitsvektor kann je nach Kontext z.B. auf Tage, Jahre oder Jahrhunderte gesetzt werden. Ursprung des Koordinatensystem ist immer die Sonne mit $\mathbf{r}_{sonne} = (0, 0, 0)$ und $\mathbf{v}_{sonne} = (0, 0, 0)$. Eine Veranschaulichung des Koordinatensystems und der Vektoren ist in Abbildung 2.1 zu sehen [15].

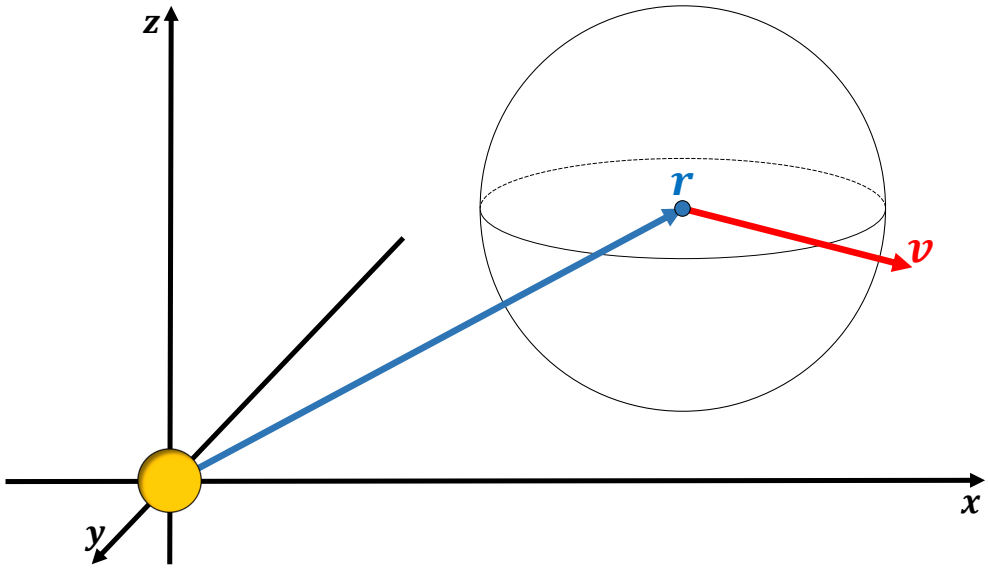


Abbildung 2.1: Genutztes Koordinatensystem mit der Sonne als Ursprung:
Jeder Himmelkörper hat innerhalb des Systems die Position **r** und eine Geschwindigkeit **v** relativ zur Sonne mit Position $\mathbf{r}_{sonne} = (0, 0, 0)$ und $\mathbf{v}_{sonne} = (0, 0, 0)$.

2.2.3 n-Körper Problem

Jeder astronomische Körper, der in dieser Arbeit betrachtet wird, verfügt über eine Masse und somit laut dem *Newtonschen Gravitationsgesetz* auch über eine eigene Gravitation. Das n-Körper Problem handelt von der Bestimmung individueller Bewegungen n astronomischer Kör-

per, die durch gegenseitige Gravitation beeinflusst sind. Es ist effizienter das n-Körper Problem im Kontext in einzelne 2-Körper Probleme aufzuspalten, wobei der Fehler im akzeptablen Bereich liegt. Das 2-Körper Problem beschreibt die Bestimmung individueller Bewegungen zweier astronomischer Körper unter ihrem gegenseitigen Gravitationseinfluss [16]. Im weiteren Verlauf dieser Arbeit werden in den Modellen durchgehend einzelne 2-Körper Probleme betrachtet.

2.2.4 Flugbahnen

Die Beschreibung interplanetarer Bahnkurven bzw. Flugbahnen astronomischer Körpers wird üblicherweise im Bezug auf die *Keplerschen Gesetze* in ein 2-Körper Problem abgebildet. Einer der beiden Körper ist der zu untersuchende, der sich, wie in Abbildung 2.2 zu sehen, auf der blauen Ellipse bewegt. Es wird von einem Basiskörper **b** ausgegangen, was hier wie in Abschnitt 2.2.2 angedeutet immer der Sonne entspricht. In Abbildung 2.2 sind zwei Ebenen zu sehen. Die Ebene B ist die Bahnebene in der die Flugbahn liegt und die Ebene R ist die sogenannte Referenzebene. Für R wird üblicherweise eine Bezugsrichtung gewählt, sodass der Äquator der Erde in ihr liegt. Zur Beschreibung der elliptischen Flugbahn reichen insgesamt sechs sogenannte Bahnelemente. Die ersten beiden Bahnelemente beschreiben die Form der Ellipse, die nächsten drei Bahnelement die Lage der Ellipse im Raum und das letzte Bahnelement ist eine Referenz der Zeit [17]:

1. **Länge der Semi-Hauptachse** a: Dieser Wert beschreibt die generelle Größe der Ellipse. a entspricht der maximalen Strecke vom Mittelpunkt **m** zur Ellipse selbst.

2. **Exzentrizität** e: Es gilt $e \geq 0$ und e beschreibt die Form der Ellipse. Gilt $e = 0$ entspricht die Flugbahn einer Kreisbahn, bei $0 < e < 1$ entspricht die Flugbahn einer Ellipse und für Werte $e \geq 1$ ist die Flugbahn nicht mehr geschlossen und wird somit para- $(e = 1)$ oder hyperbolisch $(e > 1)$. Die einzelnen Fallunterscheidungen sind beispielhaft in Abbildung 2.3 veranschaulicht. Der Term $a \cdot e$ beschreibt zusätzlich die Distanz von Mittelpunkt **m** zum Basisobjekt **b**.

3. **Inklination** i: Die Inklination beschreibt den Winkel zwischen der Bahnebene B und der Referenzebene R.

4. **Winkel Ω des aufsteigenden Knotens**: Der aufsteigende Knoten $\mathbf{k}_{\uparrow}$ ist der Punkt in R, an dem der Körper auf der Flugbahn R aufsteigend durchschreitet. Entsprechend ist Ω der Winkel zwischen der Referenzebenen-Bezugsrichtung und dem $\mathbf{k}_{\uparrow}$.

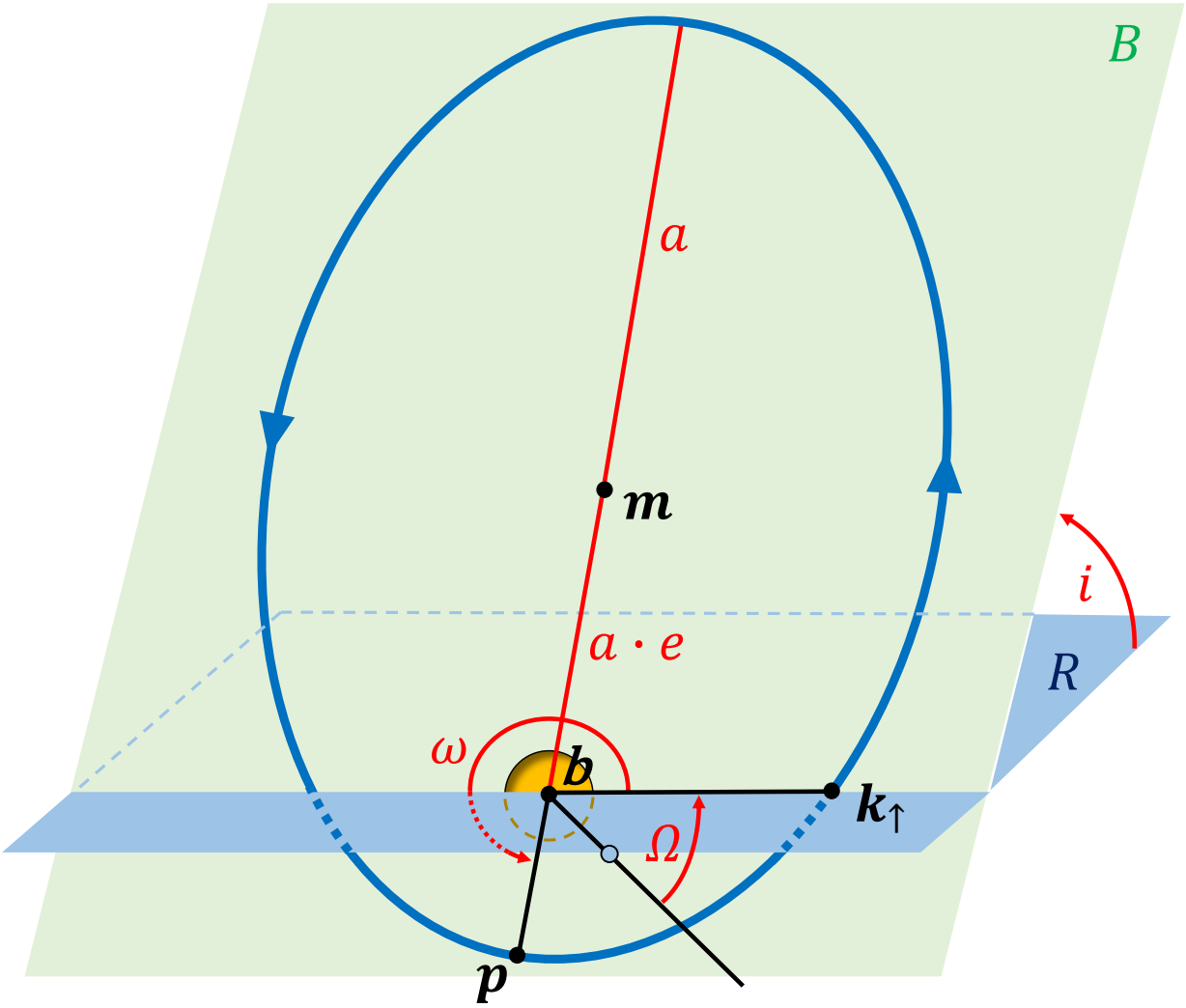


Abbildung 2.2: Bahnelemente einer Flugbahn:
Es existieren die beiden Ebenen R und B. Der Äquator der Erde liegt mit gewählter Bezugsrichtung in der Referenzebene R und die Flugbahn des Flugkörpers liegt in der Bahnebene B. **m** entspricht dem Mittelpunkt der Ellipse, **b** ist das Basisobjekt und **p** die Periapsis. Der Wert a entspricht der Semi-Hauptachse der Ellipse und e der Exzentrizität. Der Term $a \cdot e$ beschreibt die Distanz von **m** nach **b**. i steht für die Inklination, was den Winkel zwischen R und B bestimmt, Winkel Ω des aufsteigenden Knotens entspricht dem Winkel zwischen dem aufsteigenden Knoten $\mathbf{k}_{\uparrow}$ und der Referenzebenen-Bezugsrichtung und das Argument der Periapsis ω entspricht dem Winkel zwischen $\mathbf{k}_{\uparrow}$ und **p**.

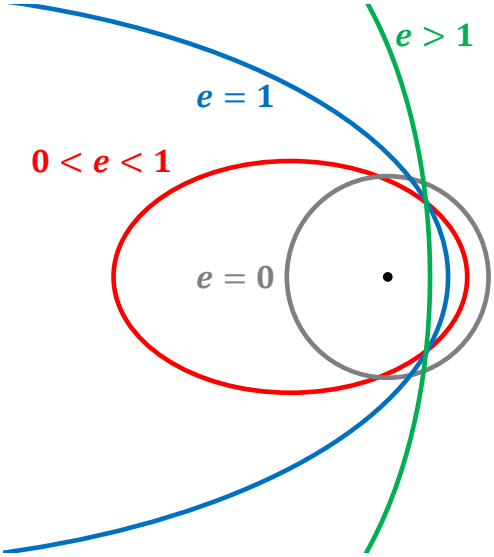


Abbildung 2.3: Fallunterscheidungen der Exzentrizität e:
Es sind vier Flugbahnen mit unterschiedlicher Exzentrizität zu sehen. Ist $e = 0$ entspricht die Flugbahn einem Kreis und gilt $0 < e < 1$ ist die Flugbahn elliptisch. Wenn $e \geq 1$ ist, entspricht es keiner geschlossenen Flugbahn mehr, sonder sie wird mit $e = 1$ para- und mit $e > 1$ hyperbolisch.

5. **Argument der Periapsis** ω: Die Periapsis **p** ist die Stelle auf der Ellipse, die die geringste Distanz zum Basiskörper **b** aufweist. ω ist der Winkel zwischen $\mathbf{k}_{\uparrow}$ und **p**.

6. **Epoche** T: Damit wird ein möglicher Zeitpunkt beschrieben, an dem der betrachtete Körper **p** durchkreuzt. Die Flugbahn ändert sich in der Realität zu jedem möglichen Zeitpunkt, wie schon in Abschnitt 2.2.3 angesprochen, durch den Gravitationseinfluss von N astronomischen Körpern. Diese Größe ist in der Hinsicht wichtig, damit klar gestellt wird welche konkrete Flugbahn in einem Modell genutzt wird.

Diese Werte werden von Instituten wie zum Beispiel dem *Jet Propulsion Laboratoy - California Institute of Technology* gemessen und in öffentlich zugänglichen Datenbanken bereitgestellt. Mit diesen Bahnelementen ist es möglich die Position und auch Geschwindigkeit eines betrachteten Körpers zu jedem möglichen Zeitpunkt abzubilden.

2.2.5 Lambert Problem

Es wird nun davon ausgegangen, dass ein Flugkörper von Position **a** nach Position **b** fliegen soll und es modellhaft nur den starken Gravitationseinfluss der Sonne gibt. Wenn der Flugkörper direkt auf Position **b** zusteuert, wird der Flugkörper durch den Gravitationseinfluss von seinem

Ursprungsziel abgelenkt, was in Abbildung 2.4 durch Flugbahn **A** veranschaulicht ist. Das heißt, es muss ein Manöver gefunden werden, dass dem Gravitationseinfluss entgegen wirkt, sodass **b** erreicht wird. Ein Beispiel für eine solche Flugbahn ist in Abbildung 2.4 bei Flugbahn **B** zu sehen.

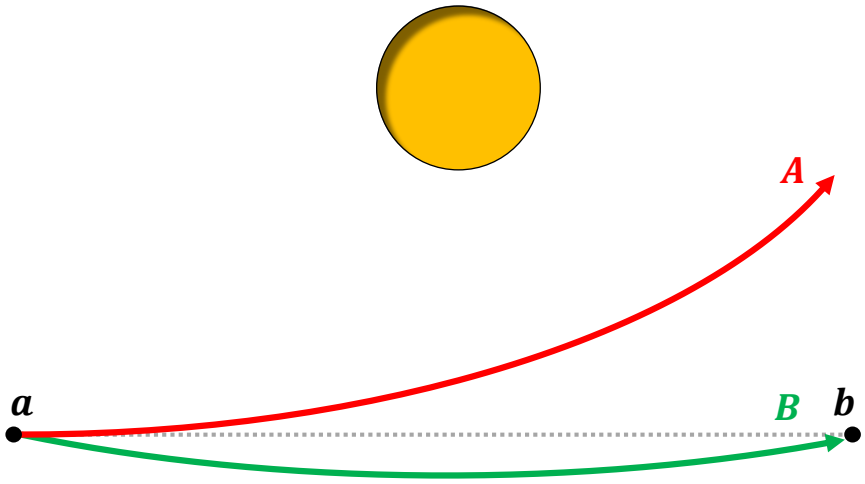


Abbildung 2.4: Gravitationseinfluss der Sonne bei Flugbahnen:
Ein Flugkörper soll unter Gravitationseinfluss der Sonne von Punkt **a** nach Punkt **b** fliegen. Wird direkt auf Punkt **b** zugesteuert wird der Flugkörper die Flugbahn **A** fliegen, mit der das Ziel **b** aber verfehlt wird. Flugbahn **B** wäre mit dem Gravitationseinfluss der Sonne die richtige Flugbahn, die das Ziel erreicht.

Allgemein gesagt ist das Lambert Problem ein 2-Körper Problem, dass sich mit der Berechnung von Geschwindigkeitsvektoren beschäftigt, wenn ein astronomischer Körper von Punkt **a** zu Punkt **b** unter dem Gravitationseinfluss eines anderen Körpers fliegen soll und genau die Zeit τ fliegt. Beim Lambert Problem kann ein beliebiger Himmelskörper übergeben werden, der als Gravitationseinfluss eingebunden wird. Das Ziel des Lambert Problems ist, passende Geschwindigkeitsvektoren zu finden, die die Abflugsgeschwindigkeit $\mathbf{v}_{\mathbf{a}\rightarrow}$ von Punkt **a** und die Ankunftsgeschwindigkeit $\mathbf{v}_{\rightarrow\mathbf{b}}$ an Punkt **b** beschreiben.

In Abbildung 2.4 war die elliptische Flugbahn **A** als mögliche Lösung zu sehen. Es kann vorkommen, dass die Distanz der Strecke, die zurückgelegt werden soll, für die vorgegebene Zeit τ zu klein oder $\mathbf{v}_{\mathbf{a}\rightarrow}$ einen ungünstigen Winkel zum Zielpunkt **b** ausweist. Dafür gibt es weitere Möglichkeiten innerhalb Lambert Problems. Eine Möglichkeit ist die Unterscheidung zwischen sogenannten *Short-Way* und *Long-Way* Flugbahnen wie in Abbildung 2.5 beispielhaft zu sehen ist.Flugbahn **S** entspricht der *Short-Way* und **L** der *Long-Way* Flugbahn. Es wird dabei unterschieden mit welchem Winkel α sich der Flugkörper insgesamt betragsmäßig dreht. Bei Drehungen mit Winkel $\alpha < 180°$ wird von einer *Short-Way* und mit Winkel $180° < \alpha <$

$360°$ von einer *Long-Way* Flugbahn gesprochen. Flugbahnen mit Winkel $\alpha = 180°$ werden als *Hohmann Transfer* bezeichnet und dabei sind die Distanzen beider Flugrichtungen gleich. Dieser Transfer wird immer als energetisch bestes Manöver angesehen [18].

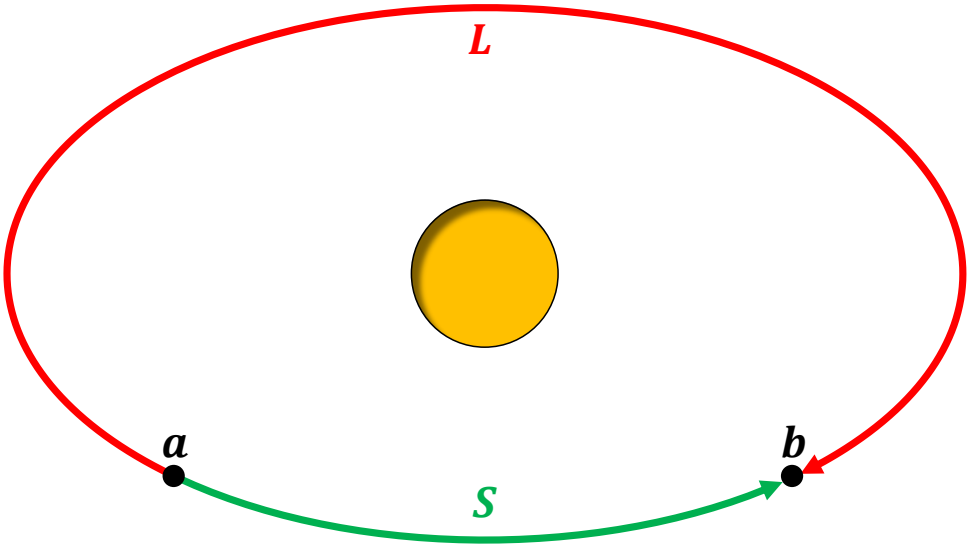


Abbildung 2.5: Short- und Long-Way beim Lambert Problem:
Ein Flugkörper soll von Punkt **a** nach Punkt **b** fliegen. Der Flugkörper kann entweder die *Short-Way* Flugbahn **S** oder die *Long-Way* Flugbahn **L** abfliegen.

Eine andere Möglichkeit die Flugbahn zu verlängern oder ungünstige $\mathbf{v}_{\mathbf{a}\rightarrow}$ zu verarbeiten ist die Einbindung von sogenannten *Revolutionen*. Dabei sind Drehwinkel $\alpha > 360°$ zulässig wenn im Abflugspunkt eine Gravitation herrscht. Ein Flugkörper kann also um eine Position **a** *schwingen* um seine Flugbahn zu verlängern, was beispielsweise in Abbildung 2.6 zu sehen ist. Im Beispiel ist eine *Revolution* zu sehen, d.h. der Drehwinkel liegt im Bereich $360° < \alpha < 720°$. Allgemein sind mehrere Umkreisungen von **a** möglich, sodass n *Revolutionen* mit Drehwinkel $n \cdot 360° < \alpha < n \cdot 720°$ entstehen. Auch hier ist die Wahl der Flugrichtung (*Short-Way* und *Long-Way*) relevant. Die Kombination aus Flugrichtung und Anzahl *Revolutionen* kann also zu kurze Transferzeiten oder ungünstige Winkel zum Zielpunkt **b** kompensieren [19].

Allgemein formuliert ergibt sich folgende Definition für das Lambert Problem. Die boolsche Variable $b_{\text{sw}} \in \{\text{true}, \text{false}\}$ gibt die Richtung der Flugbahn an, d.h. ob eine *Short-Way* Flugbahn gewählt werden soll oder nicht und die Zahl $n_{\text{rev}} \geq 0$ gibt die Anzahl *Revolutionen* an:

Eingabe Positionsvektor **a**, Positionsvektor **b**, Transferzeit $\tau > 0$, *Short-Way* b_{sw}, Anzahl Revolutionen $n_{\text{rev}} \geq 0$

Ausgabe Abflugsgeschwindigkeitsvektor $\mathbf{v}_{\mathbf{a}\rightarrow}$, Ankunftsgeschwindigkeitsvektor $\mathbf{v}_{\rightarrow\mathbf{b}}$

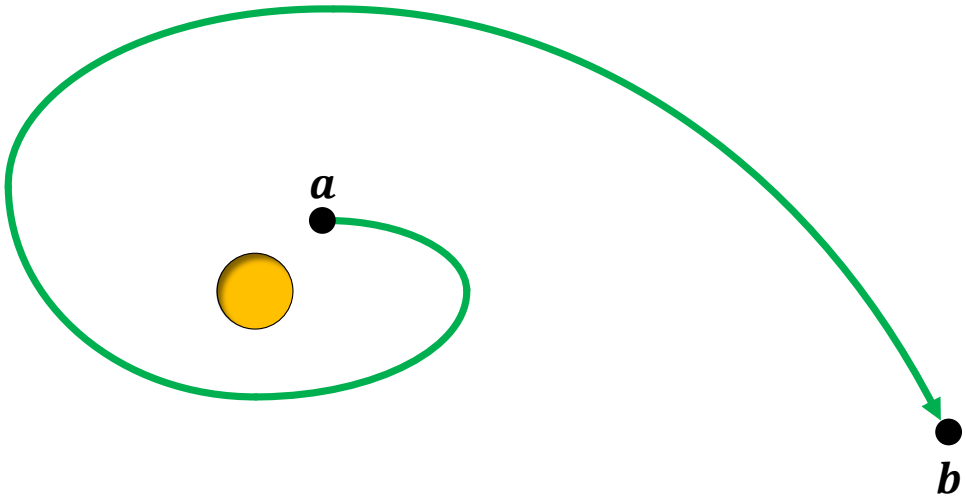


Abbildung 2.6: Beispiel für Lambert Problem mit einer Revolution:
Ein Flugkörper soll von Punkt **a** nach Punkt **b** fliegen. Durch das einmalige Umkreisen des Punktes **a** wird eine sogenannte *Revolution* um den Punkt **a** abgeflogen.

Ist τ zu klein gewählt, kann keine Lösung für diese Konfiguration von Eingaben existieren, was effektiv heißt, dass der Flugkörper nicht in der Lage ist die Strecke in der angegebenen Zeit zu fliegen. Dies kann damit zusammenhängen, dass die Distanz für die geforderte Zeit τ zu groß ist und der Flugkörper die geforderte Geschwindigkeit niemals erreichen wird, da nicht genügend Treibstoff vorhanden ist oder schneller als Lichtgeschwindigkeit geflogen werden müsste. Wenn eine Lösung gefunden wurde, ist es somit möglich Vektoren zu bestimmen, sodass die Abflugsrichtung dem Gravitationseinfluss der Sonne entgegenwirkt und unter einer bestimmten Ankunftsrichtung seine Zielposition erreicht und genau die Zeit τ für den Flug benötigt hat [20].

2.2.6 Gravity Assist

Gravity Assist bezeichnet ein Manöver in der Raumfahrt, bei der ein Flugkörper dicht an einem anderen Körper mit größerer Masse vorbeifliegt. Durch den Gravitationseinfluss des größeren Körpers ändert der Flugkörper seine Flugrichtung und -geschwindigkeit. In dieser Arbeit wird modellhaft davon ausgegangen, dass die Gravitation erst ab einer bestimmten Distanz zum größeren Körper relevant wird und innerhalb dieser Gravitationssphere konstant bleibt [15].

In Abbildung 2.7 sind jeweils zwei Beispiele mit Planeten für ein solches Manöver zu sehen. Der Flugkörper bewegt sich mit einer bestimmten Richtung abhängig von seinem Geschwin-

digkeitsvektor **v** mit der Geschwindigkeit $|\mathbf{v}|$. In den chronologisch ablaufenden Teilen 1.1) und 1.2) erfährt der Flugkörper einen Geschwindigkeitszuwachs und eine Richtungsänderung. Dies kommt zustande, da der Flugkörper durch den Gravitationseinfluss des Planeten mitgezogen und abgelenkt wird. In den Teilen 2.1) und 2.2) erfährt der Flugkörper eine Geschwindigkeitsabnahme, da hier die Gravitationskraft der Bewegung des Flugkörpers entgegenwirkt und auch in diesem Fall wird der Flugkörper abgelenkt. Insgesamt ist es im Idealfall also möglich durch die Ausnutzung von Gravitationskräften die Flugbahn und -geschwindigkeit eines Flugkörpers zu ändern, ohne Treibstoff o.ä. zu verbrauchen [21].

Die Geschwindigkeitsänderung g ergibt sich aus zwei Teilen. Der eine Teil ist der passive Einfluss durch die Gravitation des größeren Körpers und der andere Teil ist der aktive Teil durch Einwirkung mit beispielsweise Treibstoff. Das Modell, wie in [21, 15] vorgestellt, bestimmt genau diesen Teil. Das Modell sieht vor, dass ab erreichen der Gravitationssphere der Flugkörper sich bei der Position des größeren Körpers an Ort und Stelle dreht und dann die Gravitationssphere wieder verlässt. Generell muss durch den Gravitationseinfluss des größeren Körpers auch ein Mindestabstand d_{min} während des gesamten Manövers eingehalten werden, damit der Flugkörper nicht mit dem größeren Körper kollidiert oder nicht mehr in der Lage ist die Gravitationssphere wieder zu verlassen und sich eine stationäre Umlaufbahn um den größeren Körper entwickelt. Der Mindestabstand wird im Modell ebenfalls abgedeckt und es wird der tatsächliche Abstand d zurückgegeben. Insgesamt sieht das Problem dahinter wie folgt aus:

Eingabe Eingehender Geschwindigkeitsvektor $\mathbf{v}_{\rightarrow\mathbf{r}}$, Ausgehender Geschwindigkeitsvektor $\mathbf{v}_{\mathbf{r}\rightarrow}$

Ausgabe Aktive Geschwindigkeitsänderung $\Delta\mathbf{v}$, minimaler Abstand zum Planeten d

Die Berechnung beinhaltet eine Schnittpunktsuche aus Eingangs- und Ausgangsbeschleunigung im Bezug auf die Richtungsänderung, die der Flugkörper erfahren wird. Existiert kein solcher Schnittpunkt, so kann der Flugkörper trotz beliebiger Schubeinwirkung die ausgehende Geschwindigkeit $\mathbf{v}_{\mathbf{r}\rightarrow}$ nicht erreichen. Des Weiteren sind Lösungen ungültig, wenn im Nachhinein die Bedingung $d > d_{min}$ verletzt wird.

Die passive Geschwindigkeitsänderung berechnet sich aus dem Geschwindigkeitsunterschied der übergebenen Vektoren $|\mathbf{v}_{\rightarrow\mathbf{r}} - \mathbf{v}_{\mathbf{r}\rightarrow}|$. Insgesamt ergibt sich für g also

$$g = |\mathbf{v}_{\rightarrow\mathbf{r}} - \mathbf{v}_{\mathbf{r}\rightarrow}| + \Delta\mathbf{v} \tag{2.1}$$

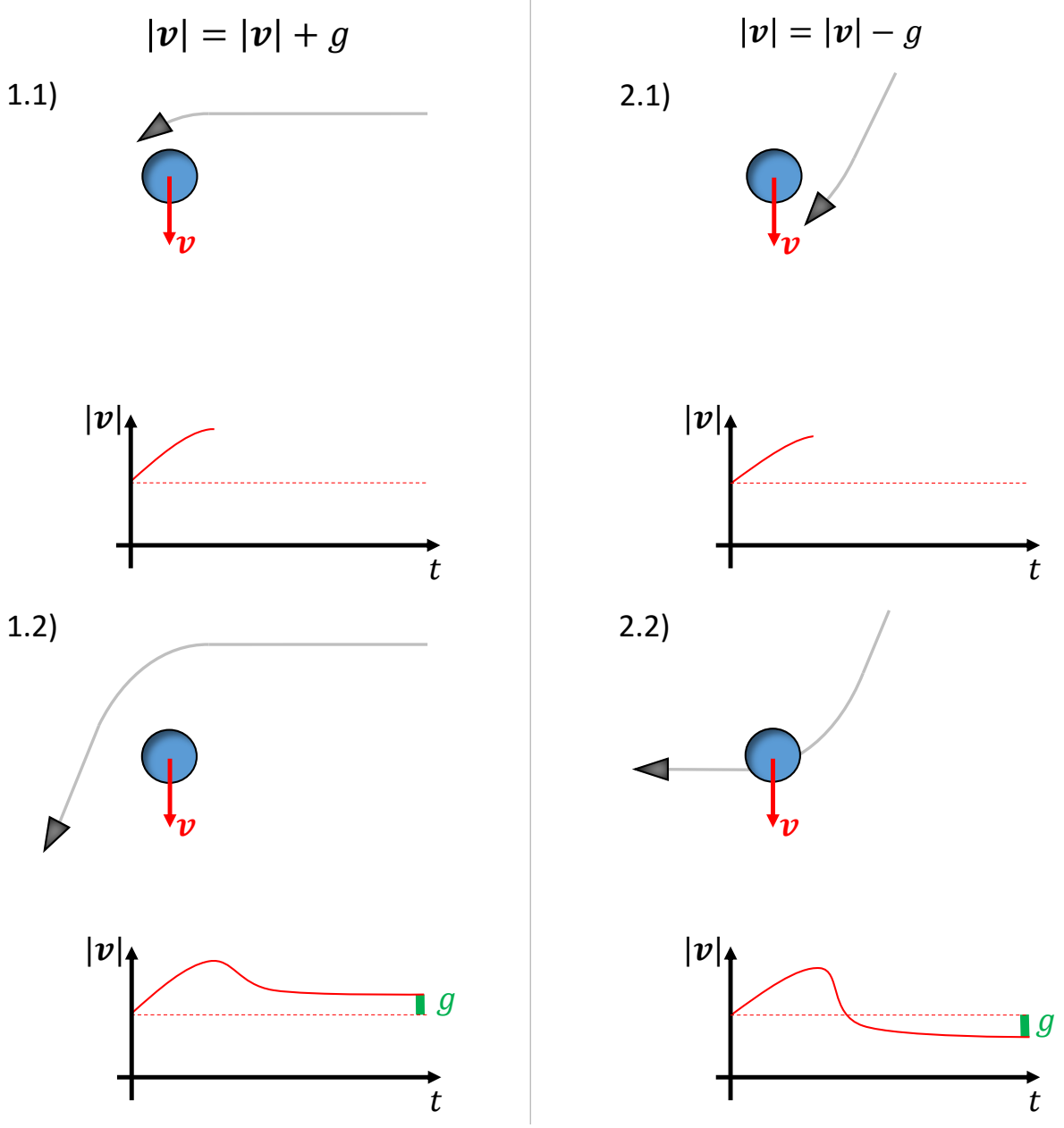


Abbildung 2.7: Gravity Assist:
Die linke Spalte zeigt den Ablauf des Gravitationseinflusses mit Geschwindigkeitszuwachs des Raumschiffes durch den mitziehenden Gravitationseinfluss des Himmelskörpers. Die rechte Spalte zeigt hingegen den Ablauf des Gravitationseinflusses mit Geschwindigkeitsabnahme des Raumschiffes durch den den entgegen wirkenden Gravitationseinfluss des Himmelskörpers. In beiden Fällen entsteht eine Geschwindigkeitsänderung g und ein Ablenken der Richtung des Flugkörpers.

2.2.7 Patched Conic Approximation

Aktuell wurden nur Ausschnitte einer Flugbahn dargestellt. Patched Conic Approximation (PCA) ist eine Methode zur Bestimmung von Flugbahnen über eine Sequenz von zu besuchenden astronomischen Körpern. PCA beinhaltet die Berechnung von Lambert Problemen um Flugabschnitte zwischen einzelnen Körpern zu bestimmen, sowie die Berechnung des Verhaltens eines Flugkörpers in einer Gravitationssphere, was Gravity Assist entspricht. In dieser Methode wird davon ausgegangen, dass Gravity Assist keine Zeit benötigt und im Mittelpunkt des Körpers eine reine Rotationsbewegung macht [22].

Der prinzipielle Ablauf von PCA ist beispielhaft in Abbildung 2.8 zu sehen. Mit einem **L** markierte Pfeile entsprechen dem berechneten Flug aus dem Lambert Problem und die mit einem **G** markierten Pfeile entsprechen dem Gravity Assist. Die in Abbildung 2.8 abzufliegende Sequenz ist $\mathbf{s} = [\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}]$. Der Ablauf des Beispieles ist folgender [23, 15]:

1) Lambert Problem von **a** zu **b**

2) Gravity Assist bei **b**

3) Lambert Problem von **b** zu **c**

4) Gravity Assist bei **c**

5) Lambert Problem von **c** zu **d**

Im Allgemeinen benötigt PCA insgesamt zwei Komponenten zur Berechnung: Einmal eine Sequenz aus $n \geq 2$ astronomischen Körpern $\mathbf{s} = [k_1, k_2, \ldots, k_n]$, die vom Flugkörper abgeflogen werden sollen und einen Vektor $\mathbf{t} = (t_0, t_1, \ldots, t_{n-1})$, der die initiale Abflugzeit t_0 im julianischen Format und alle Transferzeiten $t_1, t_2, \ldots, t_{n-1} > 0$ bestimmt. Die Transferzeit t_1 beschreibt die benötigte Flugzeit vom Körper k_1 nach Körper k_2, t_2 beschreibt die benötigte Flugzeit vom Körper k_2 nach Körper k_3, usw. Die Positionen der anzufliegenden Körper können durch **t** und die Bahnelemente bestimmt werden. Der richtige Zeitpunkt der erwarteten Ankunft am Körper k_i kann durch $t_{k_i} = t_0 + \sum_{m=1}^{i-1} t_m$ berechnet werden.

Der PCA kann mit einer kontextbezogenen Zielfunktion f kombiniert werden. Die Zielfunktion f bezieht sich auf die Sequenz **s**, den gewählten Zeitvektor **t** und gibt je nach Definition z.B. den Treibstoffverbrauch oder Änderung der Masse des Flugobjekts wieder.

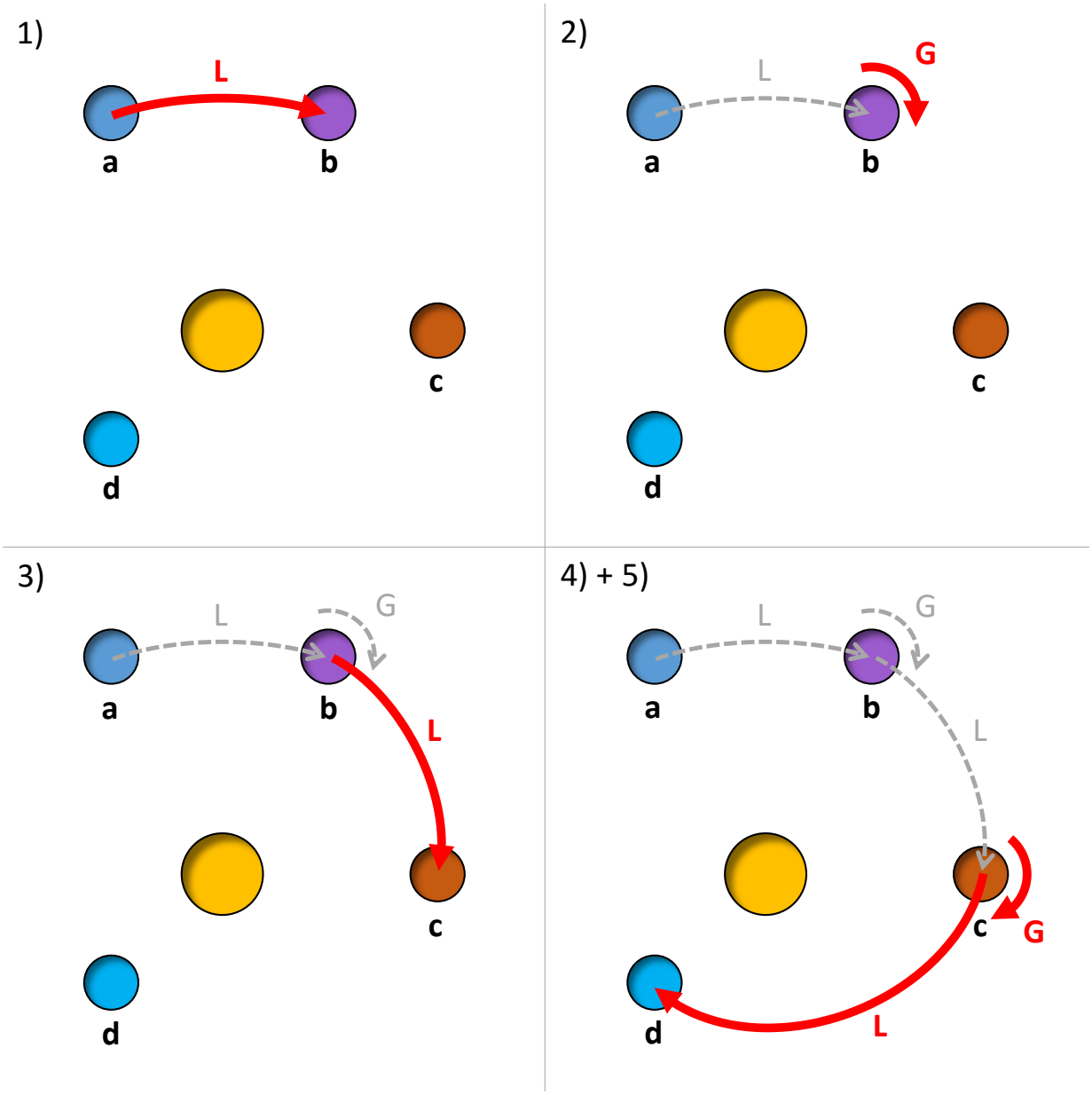


Abbildung 2.8: Beispiel für einen grundlegender Ablauf von Patched Conic Approximation: Es soll die Strecke von **a** nach **d** über **b** und **c** berechnet werden. In Schritt 1) wird die Flugbahn von **a** nach **b** mit Hilfe des Lambert Problems (L) berechnet. Darauf folgt in Schritt 2) die Berechnung des Gravity Assist (G) an **b**. Anschließend wird in 3) das Lambert Problem von **b** nach **c** berechnet. In Schritt 4) und 5) wird der Gravity Assist an **c** und abschließend das Lambert Problem von **c** nach **d** berechnet.

Allgemein formuliert ist die Systematik des PCA mit einer Zielfunktion f wie folgt:

Eingabe Flugsequenz **s**, Zeitvektor **t**, Zielfunktion f

1. Berechne alle Positionsvektoren $\mathbf{r}_{k_i}$ in Abhängigkeit von t_{k_i}
2. Berechne Lambert Probleme in Abhängigkeit der Transferzeiten $t_1, \ldots, t_{n-1}$
3. Berechne alle Gravity Assists in Abhängigkeit der erhaltenen Geschwindigkeitsvektoren aus den Lambert Problemen
4. Summiere alle aktiven und passiven Geschwindigkeitsänderungen g

Ausgabe $f(g)$

3 Global Trajectory Optimisation Competition

Die regelmäßig stattfindende *Global Trajectory Optimisation Competition* (GTOC) wurde erstmalig im November 2005 von der *European Space Agency* (ESA) initialisiert. Dieser Wettbewerb behandelt von der ESA veröffentlichte *nearly-impossible optimisation problems*, die es innerhalb eines Monats bestmöglich zu lösen gilt. Es wird nur die Problemstellung veröffentlicht und den Teilnehmern jede Freiheit zum Lösungsweg überlassen.

Abbildung 3.1 zeigt die Anzahl der Teilnehmer bezogen auf die jeweiligen abgeschlossenen Durchgänge des GTOC. Verglichen mit den ersteren Durchgängen ist trotz Schwankungen eine steigende Tendenz bei der Anzahl der Teilnehmer und das Interesse an der Domäne und dem Wettbewerb zu erkennen. Insgesamt sind über die vergangenen acht Durchgänge 134 Institute an GTOC beteiligt gewesen [24]. Der 9. Durchgang der GTOC ist aktuell in der Planungsphase und wird aller Vorraussicht Anfang 2017 starten [25].

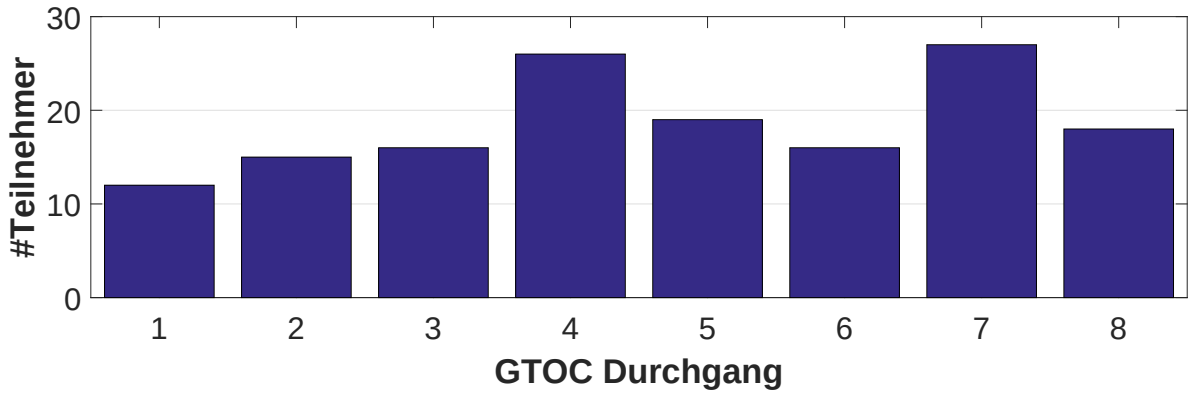


Abbildung 3.1: Anzahl der Teilnehmer bei der Global Trajectory Optimization Competition über die jeweiligen Durchgänge

3.1 *Save The Earth*

In dieser Arbeit wird das Problem des 1. Durchgangs der GTOC näher behandelt. Es wird *GTOC1 Problem* oder auch *Save The Earth* genannt. Zur Motivation: Ein Asteroid bewegt sich auf seiner Flugbahn gefährlich nahe auf die Erde zu. Es soll ein Raumschiff von der Erde aus gestartet werden, das mit dem Asteroiden kollidiert um seine Flugbahn durch den Impuls (im übertragenen Sinne als „Schwung“ vorstellbar) maximal abzulenken, damit der Asteroid sicher an der Erde vorbeifliegen kann. In Abbildung 3.2 ist es beispielhaft dargestellt. Die Ursprungs-

flugbahn ist die Flugbahn **A** und es soll durch die Kollision in etwas vergleichbares wie Flugbahn **B** oder **C** verändert werden.

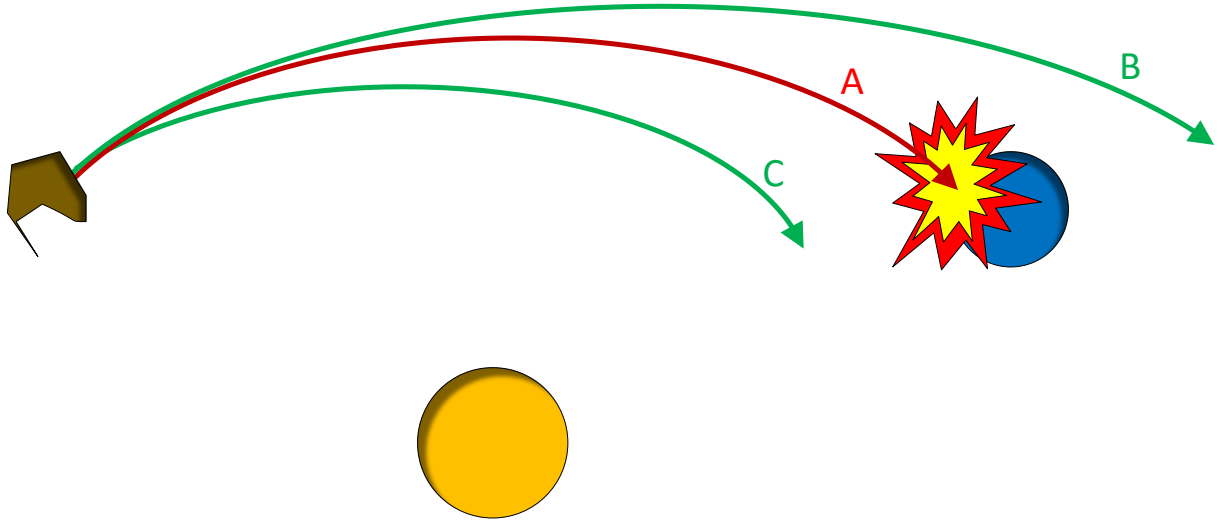


Abbildung 3.2: Motivation von *Save The Earth*:
Ein Asteroid bewegt sich auf der Flugbahn **A** gefährlich nahe auf die Erde zu. Das Ziel der Mission ist, den Asteroiden soweit abzulenken, sodass die Flugbahn des Asteroiden in eine vergleichbare wie **B** oder **C** geändert wird.

Die konkrete Problemstellung der ESA gibt die Bahnelemente des Asteroiden *2001 TW229*, Eigenschaften des Raumschiffs, einen zeitlichen Rahmen wie auch eine konkrete Zielfunktion vor. Das Raumschiff soll im Zeitintervall [55197,62502] MJD = [01.01.2010 00:00, 01.01.2030 00:00] starten und der Flug darf insgesamt nicht länger als 30 Jahre dauern. Das Raumschiff hat eine vorgegebene Schubkraft (0.05 Newton), Masse (1500 Kilogramm) und einen spezifischen Impuls (2500 Sekunden).

Die konkrete Zielfunktion dieses Problems berechnet die Aufschlagkraft F des Raumschiffs auf den Asteroiden und sieht wie folgt aus:

$$F = m_{\text{final}} \cdot |\mathbf{v}_{\text{rel}} \cdot \mathbf{v}_{\text{ast}}| \tag{3.1}$$

Die grundlegende Definition der Kraft ist laut dem *zweiten Newtonschen Gesetz* die Masse eines Objektes multipliziert mit dessen Beschleunigung. In der oben genannten Zielfunktion ist diese Definition wiedererkennbar. Der Wert m_{final} beschreibt die finale Masse des Raumschiffs bei der Kollision mit dem Asteroiden. Die ursprüngliche Masse von 1500kg kann während des Fluges durch Treibstoffverbrauch abnehmen. Folglich ist es hierbei ein indirektes Ziel auch

den Treibstoffverbrauch zu minimieren um die Masse bzw. die Aufschlagkraft weitestgehend unberührt zu lassen. Ebenfalls ist die Aufschlagstelle am Asteroiden wichtig. Der Asteroid verfügt über eine Eigenbewegung $\mathbf{v}_{\text{ast}}$, die bzgl. der Raumschiffsbewegung $\mathbf{v}_{\text{rs}}$ relativiert werden muss, was mit dem Vektor $\mathbf{v}_{\text{rel}} \in [0,1]^3$ in Gleichung 3.1 abgedeckt ist. Im Endeffekt verliert die Bewegung des Asteroiden $\mathbf{v}_{\text{ast}}$ bei ungünstiger Aufschlagstelle in der Gleichung an Bedeutung. Das heißt, dass eine relativ zur Bewegung des Asteroiden frontale Kollision bezogen auf die Aufschlagkraft F am effektivsten wäre [26].

Ein Berechnung oder eine Flugbahn wurde vom Autor *Dario Izzo* selbst in der Beschreibung der Problemstellung nicht vorgegeben. In dieser Arbeit wird zur Berechnung der Zielfunktion das Modell des MGA benutzt, das im aktuellen Stand der Forschung in vielen Veröffentlichungen ebenfalls genutzt wird. MGA benötigt, wie in Abschnitt **??** angesprochen, eine Sequenz von abzufliegenden astronomischen Körpern. Welche Auswirkungen die Wahl einer Sequenz auf das Problem selbst hat, wird im nächsten Abschnitt 3.2 näher erläutert. Es wird vorerst in diesem Abschnitt davon ausgegangen, dass eine solche Sequenz **s** der Länge n gegeben ist. Die Länge von **s** entspricht gleichzeitig der Dimension der konkreten Funktion f_s mit folgender Abbildung:

$$f_s : \mathbf{t} \mapsto F \tag{3.2}$$

Der Vektor $\mathbf{t} = (t_0, t_1, t_2, ..., t_{n-1}) \in \mathbb{R}^n$ beschreibt die im Abschnitt 2.2.5 angesprochene Abflugszeit t_0 und sämtliche Transferzeiten $t_1, t_2, ..., t_n - 1$ des Raumschiffs zwischen den n Körpern. Die Zeiten aus **t** haben wie in einigen Teilen des Abschnittes 2.2 angesprochen maßgeblich Einfluss auf die Postionen der anzufliegenden Körper, somit auch auf die Berechnung von Lambert Problemen und letztendlich auch auf die Berechnung von Gravity Assists. Somit hat der Zeitvektor insgesamt Einfluss auf die Geschwindigkeiten, die in Gleichungen 3.1 zu sehen waren. Implizit wird dadurch auch die Masse des Raumschiffs beeinflusst, was durch Treibstoffverbrauch für Beschleunigungs- oder Bremsmanöver zu erklären ist. Der Funktionswert $F \in \mathbb{R}$ der Abbildung entspricht der oben erwähnten berechneten Aufschlagkraft in Gleichung 3.1. In anderen Worten formuliert: Es sollen die Abflugzeit wie auch die Transferzeiten dahingehend optimiert werden, dass die Aufschlagkraft auf den Asteroiden *2001 TW229* maximal wird und somit handelt es sich in erster Linie um ein Maximierungsproblem. Die in dieser Arbeit genutzten Optimierungsalgorithmen suchen nach Minima und somit wird das Problem negiert. Es wird das $\mathbf{t} = (t_0, t_1, t_2, ..., t_{n-1})$ gesucht, das einen optimalen Funktionswert $f_s(\mathbf{t})$ aufweist. Damit entsteht das folgende Minimierungsproblem:

$$\arg\min_{\mathbf{t} \in \mathcal{L}} f_s(\mathbf{t}) \Rightarrow \mathbf{t} \in \mathcal{O} \tag{3.3}$$

3.2 Problemfamilie

Im vorherigen Abschnitt ist man davon ausgegangen, dass eine Flugsequenz **s** für den MGA gegeben ist, eine konkrete Zielfunktion f_s und damit ein konkretes Minimierungsproblem entsteht. Die Problemfamilie bzgl. des GTOC1 Problems beinhaltet durch die unendliche Variabilität der übergebenen Flugsequenzen unendlich viele konkrete Zielfunktionen. Dadurch entsteht ein Mixed-Integer Problem, da zu dem kontinuierlichen Optimierungsproblem mit Zielfunktion f_s eine übergeordnete diskrete Optimierungskomponente (Flugbahnsequenz) dazu kommt. Durch die Änderung des Sequenz **s** ändert sich also die Zielfunktion selbst. Einerseits bestimmt die Länge n dieser Sequenz die Dimension der Zielfunktion, da sich die Anzahl an übergebenen Transferzeiten $t_1, t_2, ..., t_{n-1}$ daran anpassen muss. Andererseits ändern sich bei gleichbleibender Länge der Sequenz **s** die Abhängigkeiten bezüglich der anzufliegenden astronomischen Körper auch wenn nur eine einzige Komponente in **s** geändert wird. Als Veranschaulichung der eben genannten Abhängigkeiten dient Abbildung 3.3.

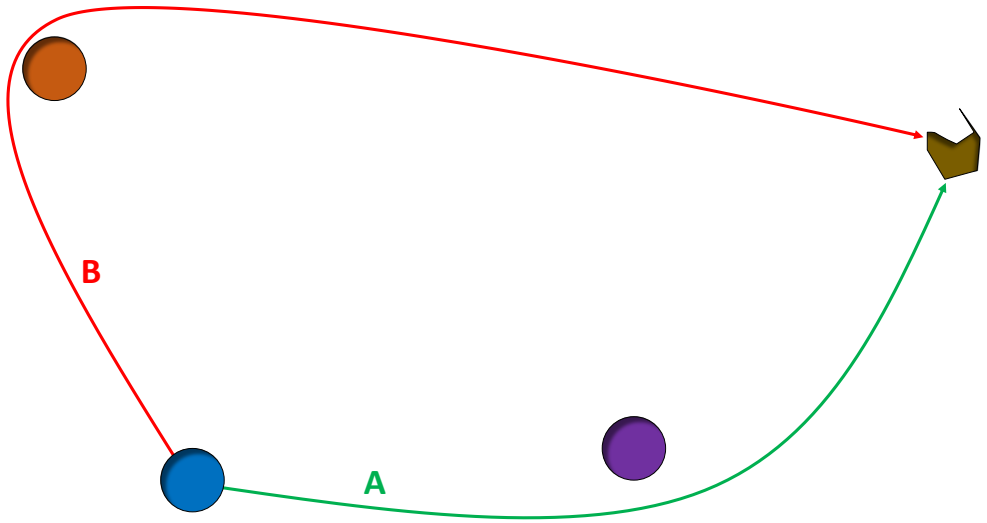


Abbildung 3.3: Wahl der Flugsequenz und die Auswirkungen:
Es sind zwei beispielhafte Flugbahnen **A** und **B** für das Raumschiff zu sehen. Das Ziel beider Flugbahnen ist jeweils mit dem Asteroiden *2001 TW229* zu kollidieren, der auf der rechten Seite zu sehen ist. Bei Flugbahn **A** muss das Raumschiff eine geringere Distanz zurücklegen als bei Flugbahn **B**.

Mit den Flugbahnen **A** und **B** soll im Endeffekt das selbe Ziel erreicht werden. Die Flugsequenzen haben zwar die selbe Länge, sind aber unterschiedlich von der Flugabfolge. Es handelt sich bei beiden Varianten um zwei 3-dimensionale Probleme mit t_0 als Abflugzeit sowie t_1 und t_2 als Transferzeiten vom Start- zum Zwischenplaneten und anschließend zum Asteroiden. Somit

ist die konkrete Zielfunktion bei beiden Varianten $f : \mathbf{t} = (t_0, t_1, t_2) \mapsto F$. Es wird nun davon ausgegangen, dass für beide Probleme immer die selben Vektoren **t** übergeben werden bzw. die Vorgabe der geforderten Zeiten zum Abflug und Transfer immer gleich sind. In dem Beispiel von Abbildung 3.3 ist erkennbar, dass die beiden Flugbahnen **A** und **B** unterschiedlich lang sind. Das lässt sich besonders durch die unterschiedliche Distanz der Planeten im Allgemeinen erklären. Bei selber Eingabe **t** muss Flugstrecke **B** insgesamt mit einer höheren Geschwindigkeit abgeflogen werden damit die vorgegebene Zeit aus **t** eingehalten werden kann. Das kann unter Umständen einen höheren Treibstoffverbrauch bedeuten, was die Masse und somit die Aufschlagkraft beeinträchtigt. Die höhere Geschwindigkeit könnte rein theoretisch aber auch durch den Gravity Assist am Zwischenplaneten erreicht werden, sodass da kein Treibstoff verbraucht werden muss. Andererseits könnte es auch sein, dass durch die sich ständig ändernden Positionen der anzufliegenden Körper aus der Sequenz eine günstige Konstellation entsteht und es sich lohnt langsamer zu fliegen. Bei Flugbahn **A** könnte das Raumschiff ggf. durch Bremsmanöver an Treibstoff verlieren, was bei Flugbahn **B** evtl. nicht der Fall sein muss. Die Variationen an möglichen Erklärungen welche der beiden Flugstrecken in Abhängigkeit zu **t** qualitativ besser im Bezug auf die Zielfunktion ist, zeigt wie unterschiedlich die einzelnen Umstände und somit die Funktionen sind. Bei selber Eingabe entstehen unterschiedliche Situationen, was sich an der Zielfunktion der jeweiligen Sequenz und Eingabe zeigt.

In dieser Arbeit wird der optimierbare, diskrete Aspekt des Mixed-Integer-Problems nicht weiter behandelt, da der zu untersuchende Ansatz HINT, wie es in Abschnitt 5.2 zu sehen sein wird, für rein kontinuierliche Probleme zugeschnitten ist.

4 Stand der Technik

Das ASP wurde ursprünglich 1975 von John R. Rice veröffentlicht [2]. Mit dem Beweis zum *No Free Lunch Theorem* [3] wurde 1997 gezeigt, dass es keinen pauschalen Algorithmus für alle Optimierungsprobleme geben kann. Somit ist das ASP in der heutigen Zeit immer noch ein relevantes Thema, obwohl das Problem schon lange bekannt ist. Es gibt in der Forschung Ansätze, die sich mit dem ASP beschäftigen, um automatisiert Algorithmen bestimmen zu können, die für jeweilige Optimierungsprobleme geeignet bzgl. Laufzeit oder Qualität der Lösung der Zielfunktion sind.

Ein populärer Ansatz ist sogenannte Algorithmen-Portfolios zu generieren. Ein solches Portfolio entspricht einer Wahrscheinlichkeitsverteilung, die vorgibt mit welcher Wahrscheinlichkeit aus einer Menge von Algorithmen einer ausgewählt werden soll. Zu Anfang ist die Wahl aus den Algorithmen uniform verteilt. Die Verteilung ändert sich je nach gewähltem Algorithmus und dessen Laufzeit und Qualität der Lösung. Damit sollen Algorithmen gefunden werden, die im Mittel eine geringe Laufzeit oder hohe Qualität der gefunden Lösung haben [27]. Der nächste Ansatz basiert auf Nutzung von Schedulern. Jeder mögliche Algorithmus erhält bestimmtes Zeitfenster um so viele konkrete Probleme zu lösen wie möglich. Auf Basis der Anzahl gelöster Probleme werden die jenigen Algorithmen in Schedulern bevorzugt, die besonders viele Probleme lösen konnten. Die Auswertung des besten Algorithmus und die darauffolgende Priorisierung im Scheduler wird mit Hilfe des k-*Nearest-Neighbor Classification* Verfahrens [28] durchgeführt. Hierbei werden Eigenschaften von Problemen mit Eigenschaften von k Problemen verglichen und so ähnliche Probleme gruppiert, sodass sie auf Algorithmen abgebildet werden können und im Endeffekt Algorithmen in den Schedulern Priorität haben, die viele konkrete Probleme in einem gegebenen Zeitfenster lösen können [29]. Deine andere Möglichkeit das ASP anzugehen ist Algorithmen nach Kriterien zu sortieren und so ein Ranking zu erzeugen. In [30] werden drei mögliche Rankings vorgestellt. Die Basis der Rankings stützt sich auf gemessene Fehlerraten nach Durchführung einer Kreuzvalidierung [31] und einer Menge verschiedener Datensätze. *Average Ranks* sortiert die Algorithmen nach dem durchschnittlichen Fehler über alle Datensätze. *Success Rate Ratios* baut auf der Idee auf die Algorithmen paarweise zu nach Fehlerraten vergleichen und nach dem Durchschnitt der Vergleiche zu sortieren. Das letzte Ranking *Significant Wins* beruht ebenfalls auf paarweises Vergleichen, aber steht hierbei eine statistische Signifikanz von 5% im Fokus. Anschließend wird wieder nach dem Durchschnitt der Signifikanzbetrachtung sortiert. In dem nächsten Ansatz werden Probleme und Algorithmen auf ein *Markow-Entscheidungsproblem* (MEP) [32] abgebildet. Ein Zustand des MEP sind die Eigenschaften des zu lösenden Problems und die Transitionen des MEP sind zur Auswahl stehende

Algorithmen. Die Wahl der Aktionen ist zufallsbedingt und die Zufallsverteilung ändert sich mit der Zeit durch ein erweiterten Ansatz des Q-Learnings [33]. Somit sollen geeignete Algorithmen bzgl. der Eigenschaften des Problems mit höherer Wahrscheinlichkeit gewählt werden [34].

Nun wird auf die Domäne der Berechnung interplanetarer Bahnkurven eingegangen. Bei der Lösung des Lambert Problems wird oft der *Lambert Universal Variable Algorithm* genutzt. Dies ist ein effizienter und simpler Algorithmus, der ein iteratives Schema verfolgt, das im Endeffekt zu einer möglichen Flugbahn (elliptisch, para-, hyperbolisch) konvergiert [35]. Die ESA hat sich ebenfalls seperat mit dem Lambert Problem auseinandergesetzt und ein Verfahren entwickelt, dass durch das Householder-Verfahren [36] Initiallösungen berechnet und anschließend mit eigens entwickeltem Iterationsverfahren das Lambert Problem dann in bis zu zwei Iterationsschritten lösen kann [37].

Es wurden Modelle entwickelt, die die erlaubten Manöver des Raumschiffs beschreiben. In allen hier folgenden, zitierten Veröffentlichungen auf der Domäne interplanetarer Bahnkurzen wird entweder das Modell *Multiple Gravity Assist* (MGA) oder *Multiple Gravity Assist with Deep Space Manoeuver* (MGA-DSM) genutzt. Beide Modelle sind gekoppelt mit der Berechnung der *Patched Conic Approximation*. Der MGA erlaubt es dem Raumschiff nur während des Gravity Assist Treibstoff zur Geschwindigkeitssteigerung oder -drosselung zu verbrennen und der MGA-DSM erlaubt hingegen zusätzlich den Treibstoffeinsatz während des Transfers zwischen Planeten, was der Berechnung von Lambert Problemen entspricht [38].

Dario Izzo, Initiator des GTOC, veröffentlichte eine Übersicht der genutzten Verfahren der damaligen Teilnehmer zur Lösung des GTOC1 Problems [39]. Im Folgenden wird auf die Verfahren einiger Teilnehmer eingegangen, die mit ihren Lösungen eine höhere Platzierung erreicht haben. Das Gewinner-Institut *Jet Propulsion Laboratory* (JPL) entwickelte auf Basis von Erfahrungen die zu untersuchenden Flugsequenzen. Um Treibstoffverbrauch zu mindern, hat die JPL Sequenzen gewählt die anfangs die nah befindlichen Planeten Erde und Venus oft abfliegen um über die Gravity Assists Geschwindigkeit zu generieren. Diese Sequenzen wurden mit der von der JPL entwickelten Optimierungstool *STOUR-LTGA* [40], im Grunde ein automatisiertes Grid-Search-Verfahren speziell für die Bestimmung von Flugbahnen entwickelt, untersucht und optimiert. Damit entstanden erste Lösungen, die zur weiteren Untersuchung dem Optimierungstool *MALTO*, auf Basis eines Verfahrens von Sims und Flanagan [41], übergeben wurden [42]. Auch das der nächste Teilnehmer *GMV Innovating Solutions* nutzt Optimierungstools. Im ersten Schritt haben die Experten eine geeignete Sequenz entwickelt, die anschließend in drei Teile verlegt wurde. Jeder dieser Teile wurde einzelnt für sich mit Hilfe eines selbst entwickelten, interaktiven Tools *MATLAB Interactive TRAjectory DESign* (MITRADES) [43] von den

Experten analysiert. Sobald die einzelnen Teile durch MITRADES als hinreichend gut erachtet wurden, trug man die einzelnen Teile zusammen und die gesamte Trajektorie wurde mit Hilfe eines Verfahrens auf Basis von *Genetischen Algorithmen* [44], optimiert [45]. Das Institut *DEIMOS Space* hat einen anderen Ansatz verfolgt. Als erstes wurden für ausgewählt Sequenzen Ballistik-Analysen durchgeführt. Entsprechend wurden dadurch Lambert Probleme vor den Gravity Assists berechnet. Auf Basis der Daten wurden die Sequenzen samt bestimmten Eigenschaften des Flugbahn nach den Kriterien beste Lösung, Treibstoffverbrauch und Flugzeit aussortiert. Für die so entstandenen Flugbahnen wurden anschließend mit Hilfe eines *Local Constraint Optimisers* [46] die dazu passenden Gravity Assists bestimmt, sodass eine vollständige Flugbahn entsteht [47]. Als das letzte Verfahren mit hochrangiger Lösung im Wettbewerb, die veröffentlicht wurde, gilt die von den Teilnehmern *Centre National d'Etudes Spatiales*. Sie haben ihre Lösung in zwei Schritten bestimmt. Im ersten Schritt benutzten sie das nicht-lineare Simplex-Verfahren von Neader und Mead [48] um dadurch Lösungen zu finden, die als für den zwei Schritt dienen. Im zweiten Schritt wurden die vorher bestimmten Lösungen als Initialwerte numerischen Verfahren, wie z.B. dem *Newton-Verfahren*, übernommen und so die vorher gefundenen Lösungen weiter verbessert [49].

Um einen allgemeinen Ansatz für die Domäne interplanetarer Bahnkurven zu finden, wurden von der ESA verschiedene Optimierungsalgorithmen im Bezug auf solche Probleme evaluiert um zu prüfen, ob es einen pauschalen Algorithmus für diese Art von Problemen gibt und so das ASP hier nicht problematisch ist. Auch das GTOC1 Problem wurde hierbei untersucht. Insgesamt wurden dabei die Optimierungsalgorithmen *Differential Evolution*, *Particle Schwarm Optimization*, *Genetischer Algorithmus*, *Adaptive Simulated Annealing* [50], *GLOBAL* [51] und *COOP* [52] untersucht. Dabei ergab sich, dass es für die verschieden Probleme keinen pauschalen Algorithmus für alle Probleme dieser Art gibt [6]. Desweiteren wurde für verschiedene Probleme der Domäne das Verfahren *Gravity Assist Space Pruning* (GASP) entwickelt. GASP bestimmt durch Vergleich von möglichen Abflugs- und Ankunftszeiten des Raumschiffs an verschieden Planeten, welcher der Flugzeiten überhaupt realistisch sind. Dadurch wird $\mathcal{L}$ systematisch verkleinert, sodass ein Optimierungsalgorithmus weniger Raum zum suchen haben wird [53].

5 Optimierungsverfahren

Es folgt die Beschreibung zweier Optimierungsverfahren, die in dieser Arbeit gegenübergestellt werden. Diese Optimierungsverfahren, *Direct Learning* (DL) und HINT, beschäftigen sich mit dem ASP, d.h. dem Problem der Wahl eines Algorithmus passend zum zu lösenden Problems. Beide Verfahren sollen bei Eingabe einer Flugsequenz **s** des GTOC1 Problems den passenden Algorithmus aus der Menge von verfügbaren Algorithmen **A** finden, mit dem das durch die Flugsequenz beschriebene konkrete Problem gelöst wird. Desweiteren ist bei DL und HINT durch Regressionsmodelle jeweils ein Lernmechanismus implementiert, der lernt wie gut welcher Algorithmus tatsächlich funktioniert hat. Dazu werden Informationen nach Ausführen des bestimmten Algorithmus in die Lernmechanismen eingepflegt. Das Zurückführen von Informationen nach Lösung des Problems mit beliebigen Algorithmus ist der Schlüssel zum Erfolg eines solchen Lernmechanismus. Über die Zeit werden diese Informationen stehts aktualisiert, sodass intern insgesamt ein Modell der Problemfamilie entsteht, dass auf den bestmöglichen Algorithmus abgebildet werden kann. Das Optimierungsverfahren sollte sich im Optimalfall nach einer gewissen Lernphase immer für einen effizienten bzw. qualitativen Algorithmus entscheiden.

Durch die Übergabe einer Flugsequenz **s**, die im Endeffekt eine konkrete Funktion f_s mit sich bringt, lernen die Optimierungsverfahren hier nicht die Funktionen selbst, sondern sie lernen die Problemfamilie einzelner GTOC1 Probleme, die im Abschnitt 3.2 näher angesprochen wurde. Somit wird versucht Wissen von konkreten Problemen auf andere konkrete Probleme der selben Problemfamilie zu übertragen.

5.1 Direct Learning

Dieser Ansatz ist in Abbildung 5.1 dargestellt. Das DL beinhaltet einen Teil mit dem Namen *Adaptives Mapping* (AM), der allgemein gesagt eine Problemfamilie lernt und entsprechend die Güte von b verfügbaren Algorithmen samt gewählten Parametern bestimmt. Es ist also möglich einen Algorithmus mehrfach vorkommen zu lassen, die sich in den konfigurierbaren Parametern unterscheiden. Es wird eine Flugsequenz **s** übergeben. Auf Basis dieser Information werden die mögliche Anzahl an benötigten Iterationsschritten und der erwartete Funktionswert der Zielfunktion geschätzt. In der Abbildung entspricht dies der Menge **A** aus Tupeln $\mathbf{a}_i$, die alle o.g. Informationen zu dem jeweiligen Algorithmus enthalten.

$$\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n\},$$
$$\mathbf{a}_i = (\text{Anzahl Iterationen}, \text{bester Funktionswert})$$

Die Menge an Algorithmen kann nun nach der Anzahl an Iterationen oder der besten erwarteten Lösung sortiert werden. Ist der Algorithmus bestimmt worden, der das Kriterium als besten Algorithmus erfüllt (minimale Anzahl an Iterationen oder bester Funktionswert), wird dieser auf das konkrete Problem, welches durch **s** beschrieben ist, ausgeführt. Nach der Ausführung wird die tatsächliche Anzahl Iterationen und die gefundene Lösung des Problems in das DL eingepflegt. Bei der nächsten Eingabe einer Flugsequenz fließen diese Informationen in die nächste Entscheidung mit ein.

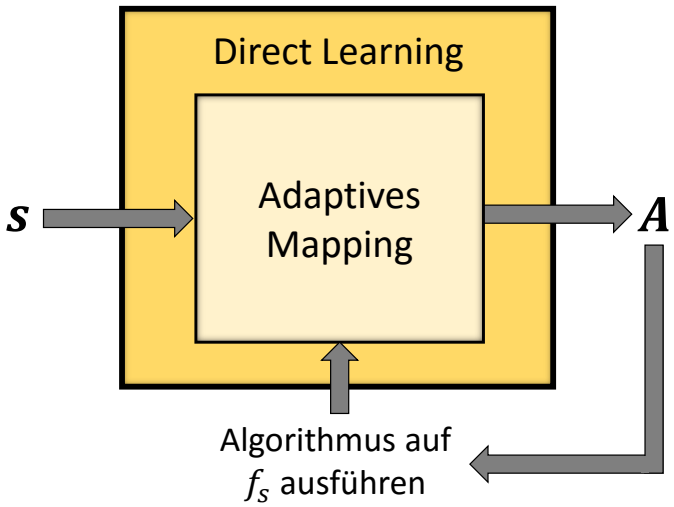


Abbildung 5.1: Übersicht *Direct Learning*:
Direct Learning beinhaltet die Mechanik *Adaptives Mapping* (AM). AM erhält eine Flugsequenz **s** als Eingabe und gibt die Menge **A** mit Informationen zu allen Verfügbaren Algorithmen bzgl. geschätzter Laufzeit und Lösung der konkreten Funktion f_s aus. Es wird der vom Kontext abhängig geeignetste Algorithmus gewählt und auf f_s ausgeführt. Die Information über die benötigte Laufzeit und Lösung der Funktion wird für die nächste Schätzung mit einbezogen.

Wie das AM im Detail funktioniert, ist in Abbildung 5.2 zu sehen. Jeder der möglichen anzufliegenden Körper wird in **s** übergeben, sodass die Eingabe insgesamt k Elemente bzw. anzufliegende Körper beinhaltet. Die Länge von **s** bestimmt implizit die Dimension der Zielfunktion f_s. Für jede mögliche Kombination aus Dimension k und b verfügbaren Algorithmen, existiert ein Regressionsmodell. Aus diesen einzelnen Regressionsmodellen werden dann die o.g. Werte für jeden Algorithmus geschätzt. Die Ausführung des ausgewählten Algorithmus sorgt für neue Informationen, die dann in das entsprechende Regressionsmodell des Algorithmus passend zur Dimension k einfließen. Dadurch bilden die einzelnen b Regressionsmodelle die Abbildung $\mathbf{s} \mapsto \mathbf{a}_i$ immer genauer ab. Wenn ein Algorithmus ausgeführt wird, der eigentlich ungeeignet ist,

wird das Regressionsmodell unter Hinzunehmen des neuen Datenpaares antrainiert, was dazu führt, dass für den Algorithmus mit geringerer Wahrscheinlichkeit gute Werte geschätzt werden und umgekehrt.

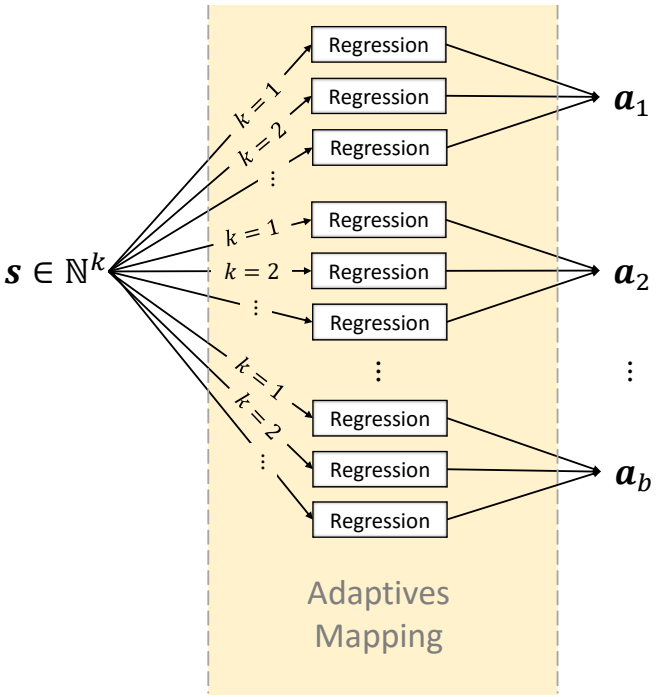


Abbildung 5.2: Detailansicht *Direct Learning*:
Es sind hier ebenfalls die Eingabe **s** und die Ausgabe **A**, dargestellt als $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_b$, zu sehen. Je nach Länge k von **s** (was der Problemdimension von f_s entspricht) werden die entsprechenden Regressionen gewählt, die Schätzungen über die Laufzeit und die Lösung der konkreten Funktion für jeden einzelnen Algorithmus $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_b$ berechnen und ausgeben sollen.

5.2 Hint-based Optimisation

Die oben genannten Regressionsmodelle kommen in diesem Ansatz ebenfalls vor, allerdings in einem anderen Zusammenhang. Die Schätzung der Güte eines Algorithmus hängt hier nur indirekt von der Eingabe **s** ab, wie es in Abbildung 5.3 zu sehen ist. Der Weg von der Flugsequenz **s** zu den Schätzungen der Güte der Algorithmen geschieht hier mit Hilfe der Bestimmung von Eigenschaften $(p_1, p_2, \ldots, p_m)$ des Problems, was als Eigenschaftenvektor $\mathbf{p} \in \mathbb{R}^m$ definiert ist. Der Vektor **p** wird im ersten Schritt von dem AM bestimmt. Das die Abbildung des AM auf die Algorithmen hier hängt nicht mehr von **s** direkt ab, sondern von dem Eigenschaftenvektor **p**. Nachdem der Algorithmus ausgewählt und ausgeführt wurde, werden nun Informationen nicht

nur in das AM eingepflegt, sondern auch in den ersten Teil mit dem Namen *Eigenschaften bestimmen*.

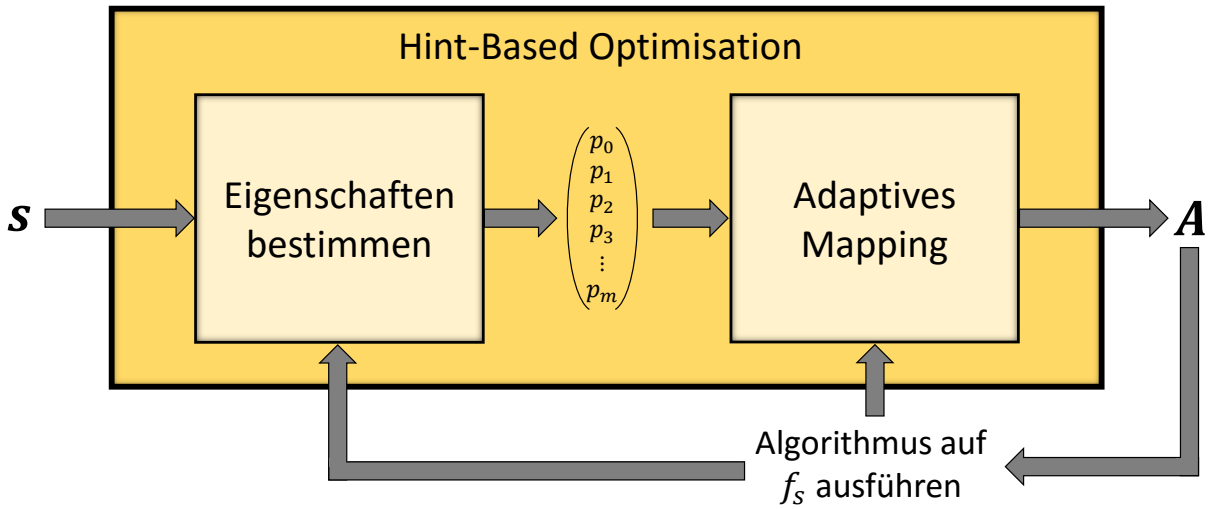


Abbildung 5.3: Übersicht Hint-Based Optimisation:
Hint-Based Optimisation beinhaltet die Mechaniken *Eigenschaften bestimmen* und *Adaptives Mapping* (AM). Die Eingabe der Flugsequenz **s** wird übergeben um einen Eigenschaftsvektor $(p_1, p_2, p_3, \ldots, p_m)$ der Länge m zu bestimmen. AM erhält statt **s**, wie bei *Direct Learning*, nun diesen Vektor als Eingabe und gibt die Menge **A** mit Informationen zu allen Verfügbaren Algorithmen bzgl. geschätzter Laufzeit und Lösung der konkreten Funktion f_s aus. Es wird der vom Kontext abhängig geeignetste Algorithmus gewählt und auf f_s ausgeführt. Die Information über die benötigte Laufzeit und Lösung der Funktion wird für die nächsten Schätzungen in beiden Mechaniken mit einbezogen.

Der Eigenschaftsvektor verhilft dem Ansatz Wissen über die Problemfamilie zu generieren und später in separate Regressionsmodelle einfließen zu lassen. Dazu existiert die folgende Matrix S:

$$S = \begin{pmatrix} \mathbf{x}_0 & \cdots & \mathbf{x}_m \\ f_s(\mathbf{x}_0) & \cdots & f_s(\mathbf{x}_m) \end{pmatrix} \tag{5.1}$$

Diese Matrix beschreibt pro Spalte Paare $\big(\mathbf{x}_i, f_s(\mathbf{x}_i)\big)$ von Punkten $\mathbf{x}_i$ im Suchraum mit dazugehörigem Funktionswert $f_s(\mathbf{x}_i)$. Die Paare sind nach Funktionswerten sortiert, sodass $f_s(\mathbf{x}_0) \leq \ldots \leq f_s(\mathbf{x}_m)$ gilt. Die einzelnen Eigenschaften p_i von **p** werden folgendermaßen berechnet:

$$p_i = \frac{f_s(\mathbf{x}_i) - f_s(\mathbf{x}_0)}{|\mathbf{x}_i - \mathbf{x}_0|}, (\mathbf{x}_i, f_s(\mathbf{x}_i)) \in S, i \geq 1 \tag{5.2}$$

Also, beschreibt der Eigenschaftsvektor **p** mit jeder seiner Komponenten die Steigung von m Punkten $\mathbf{x}_1, \ldots, \mathbf{x}_m$ im Suchraum zur besten Lösung $\mathbf{x}_0$. Die Anzahl m an Eigenschaften in **p** ist frei wählbar. Die Eigenschaften, die hier bestimmt werden, sind invariant gegenüber Translation, Rotation und Skalierung, sodass der Vorteil dabei ist, dass damit Erkenntnisse aus konkreten Problemen auf andere übertragen werden können.

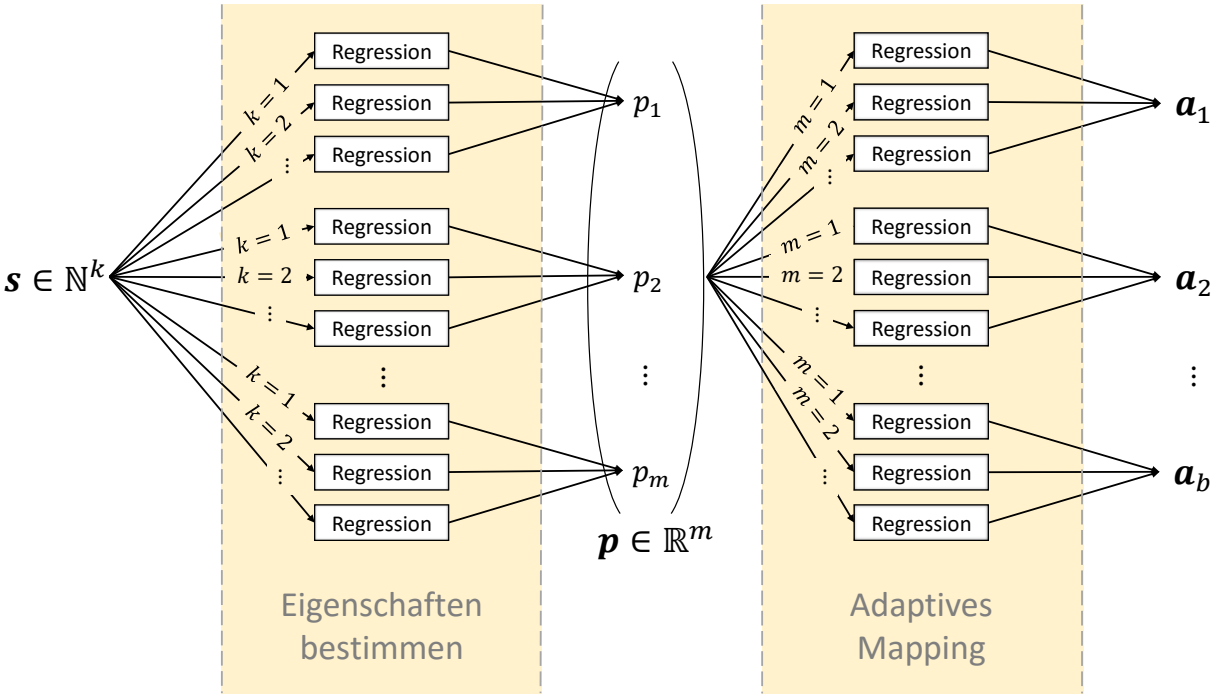


Abbildung 5.4: Detailansicht Hint-Based Optimisation:
Es sind hier ebenfalls die Eingabe **s** und die Ausgabe **A**, dargestellt als $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_b$, zu sehen. Je nach Länge k von **s** (was der Problemdimension von f_s entspricht) werden die entsprechenden Regressionen gewählt, die Eigenschaften $p_1, p_2, \ldots, p_m$ bestimmen. Daraus ergibt sich ein Eigenschaftsvektor **p** der Länge m. Darauf werden wieder Regressionen in abhängigkeit von m gewählt, die Schätzungen über die Laufzeit und die Lösung der konkreten Funktion für jeden einzelnen Algorithmus $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_b$ berechnen und ausgeben sollen.

Das detaillierte Vorgehen von HINT ist in Abbildung 5.4 zu sehen. Die Flugsequenz **s** wird dem ersten Teil übergeben. Pro Element des Eigenschaftsvektors wird bzgl. der Länge von **s** bzw. der Dimension des konkreten Problems k ein Regressionsmodell zum Schätzen einer Eigenschaft genutzt. Somit existiert hier m-mal die Abbildung $\mathbf{s} \mapsto p_i \in \mathbb{R}$. Ist jede einzelne Eigenschaft bestimmt worden, wird **p** in den zweiten Teil übergeben. Nun wird **p** auf die Algorithmen abgebildet. In Abhängigkeit der Anzahl an Eigenschaften bzw. der Länge von **p** wird für jeden Algorithmus ein Regressionsmodell genutzt um die Anzahl benötigter Iterationen und die beste Lösung eines Algorithmus zu schätzen. Wurde der entsprechende Algorithmus ausgewählt und ausgeführt, wird entsprechend die Matrix S generiert und **p** mit der Berechnung aus Gleichung 5.2 bestimmt und die Anzahl der benötigten Iterationen sowie der beste Funktionswert betrachtet. Das Regressionsmodell des ausgewählten Algorithmus wird im zweiten Teil, dem AM, bzgl. des Zusammenhangs $\mathbf{p} \mapsto \mathbf{a}_i$ aktualisiert. Im ersten Teil *Eigenschaften bestimmen* werden die m zur Dimension k passenden Regressionsmodelle bzgl. des Zusammenhangs $\mathbf{s} \mapsto p_i$ aktualisiert. Zusammengefasst besteht hier die Kette an Abbildungen aus m-mal $\mathbf{s} \mapsto p_i$ und anschließend b-mal $\mathbf{p} \mapsto \mathbf{a}_i$.

Im Vergleich zum DL muss HINT bei jeder Aktualisierung statt nur einem insgesamt $m + 1$ Regressionsmodelle aktualisieren und sämtliche Eigenschaften von **p** berechnen. Der Laufzeitunterschied des Lernprozesses macht DL im Vergleich zum HINT attraktiver, da der Lernprozess über die Regressionen schneller durchgeführt wird. Allerdings nutzt DL keinerlei Informationen über die Eigenschaften der konkreten Funktion selbst, sodass dies ein Faktor sein kann, dass HINT mit der Zeit den für jedes konkrete Problem passenden Algorithmus bezüglich seiner Güte findet.

6 Evaluation von Hint-based Optimisation

In diesem Kapitel wird HINT hinsichtlich seiner Tauglichkeit gegenüber der Problemfamilie GTOC1 evaluiert. Es wird mit der Erläuterung der Systematik der experimentellen Analyse begonnen. Zu Beginn der Analyse wird ein Ranking bezüglich der Qualität gefundener Lösungen einzelner, genutzter Optimierungsalgorithmen aufgestellt. Auf Basis dieses Rankings werden die Ergebnisse von HINT und DL bezüglich ihrer Tauglichkeit bei dieser Problemfamilie zusammengetragen und anschließend diskutiert.

6.1 Systematik der Analyse

In der Evaluation werden die Algorithmen, die in Abschnitt 2.1.2 vorgestellt worden sind, für die beiden Optimierungsverfahren genutzt. Es wird pro Algorithmus ein Satz an Konfigurationsparametern festgelegt, sodass sechs Algorithmen samt Konfigurationsparametern in die Evaluation einfließen. Es werden für die Untersuchung folgende Parameter der Optimierungsalgorithmen festgelegt:

- Die **Zufallssuche** ist parameterfrei.
- **Hill Climbing** (HC) arbeitet bei der Suche nach einem neuen Nachbarn mit einem Radius der Länge von 10% der Größe des Lösungsraums $\mathcal{L}$.
- **Simulated Annealing** (SA) nutzt eine Starttemperatur beziehungsweise -sprunggröße von 10% der Größe des Lösungsraums $\mathcal{L}$.
- **Hooke-Jeeves Algorithmus** (HJ) hat als anfängliche Distanzlänge 100% der Größe des Lösungsraums $\mathcal{L}$.
- **Differential Evolution** (DE) wird mit der Populationsgröße $4.4n^{1.23}$ abhängig von Problemdimension n ausgeführt. Die Gewichtung der Mutation ist 0.5 und die des Crossovers 1.0. Somit wird der Crossover vollständig übersprungen [54].
- **Particle Swarm Optimisation** (PSO) bekommt die Gewichtung beider Anziehungskräfte c_p und c_g ist jeweils $\frac{1}{2} + \ln 2$ und der Beschleunigungsfaktor λ wird mit $\frac{1}{2 \ln 2}$ gewichtet [55]. Allerdings wird die Populationsgröße auf 10 statt auf 40 gesetzt.

Die oben genannten Algorithmen werden jeweils auf einzelne konkrete Probleme aus der Problemfamilie GTOC1 angesetzt. Die konkreten Funktionen ergeben sich aus zufällig generierten

Flugsequenzen. Jeder der Flugsequenzen enthält im ersten Element die Erde und als letztes Element den Asteroiden *2001 TW229*. Die restlichen Elemente sind aus der Menge {Venus, Erde, Mars, Jupiter, Saturn} und werden jeweils zufällig auf diese Elemente verteilt, wobei zwei nebeneinander liegende Elemente nicht gleich sein dürfen. Die Menge an konkreten Problemen ist für alle Algorithmen gleich um sie später vergleichbar machen zu können. Alle konkreten Probleme haben eine Problemdimension von 8.

Für jedes konkrete Problem, das aus einer der Zufallsflugbahnen entsteht, wird jeder der in Abschnitt 2.1.2 genannten Algorithmen ausgeführt und anschließend jeweils genau für dieses konkrete Problem miteinander verglichen. Es werden also Zufallsflugbahnen generiert und für jedes dieser konkreten Probleme eine Rangliste erstellt, die die Algorithmen nach der Qualität der Lösung nach einer bestimmten Anzahl an Funktionsauswertungen sortiert. Somit entstehen einzelne Ranglisten für jedes dieser 8-dimensionalen konkreten Probleme.

Nachdem die Ranglisten erstellt worden sind, werden die in Kapitel 5 vorgestellten Optimierungsverfahren ausgeführt. Ziel dabei ist, dass geprüft wird, ob sich die beiden Verfahren für den jeweils besten Algorithmus aus den einzelnen Ranglisten entscheiden. Im Endeffekt soll auch gegenübergestellt werden, ob der im HINT vorkommende Ansatz des Untersuchens von Eigenschaften der Funktion von Vorteil ist und ob sich DL öfter als HINT für den bezüglich der Rangliste besten Algorithmus entscheidet.

6.2 Ranking

In diesem Abschnitt werden Rankings aufgestellt, die hier genutzte Optimierungsalgorithmen nach der Qualität seiner gefundenen Lösungen in der Problemfamilie GTOC1 sortiert. Dieses Ranking dient anschließend dazu zu prüfen, ob sich die Optimierungsverfahren DL und HINT jeweils für den nach dem Ranking passenden Algorithmen entscheiden.

Es wurden 250 zufällige Flugsequenzen generiert, sodass bis zu 250 unterschiedliche konkrete Probleme aus der Problemfamilie GTOC1 entstehen. Jeder der sechs Algorithmen, die in der Evaluation involviert sind, liefen bis 1000 Funktionsauswertungen des jeweiligen konkreten Problems berechnet wurden. Somit führte jeder Algorithmus $250 \cdot 1000 = 250000$ Funktionsevaluationen auf den zufällig bestimmten konkreten Problemen aus und suchten in jedem der 250 Durchläufe nach der bestmöglichen Lösung.

In Abbildung 6.1 sind die sechs Algorithmen gegenübergestellt. Auf der X-Achse der jeweiligen Darstellungen sind die Sequenzen, die durchlaufen wurden, dargestellt und auf der Y-Achse ist der prozentuale Anteil der Lösung im Vergleich zur besten Lösung der sechs Algorithmen der jeweiligen Sequenz zu sehen. Zum Beispiel entspricht die 200 auf der X-Achse der Sequenz

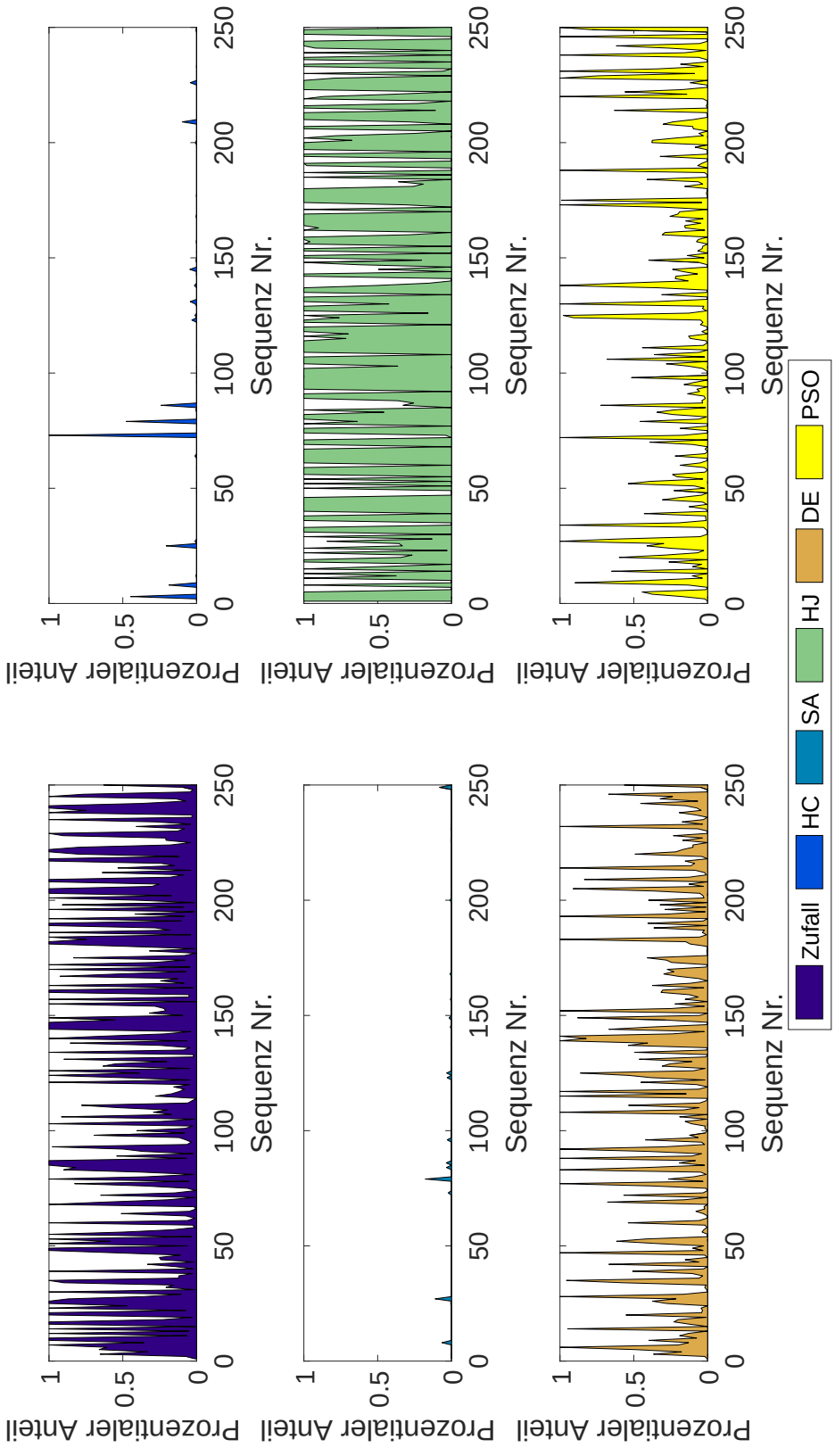


Abbildung 6.1: Übersicht der gemessenen Daten für das Ranking:
Zu sehen sind sechs Plots, die je für einen Algorithmus stehen. An der X-Achse sind zufällig generierte Flugsequenzen, die konkreten Problemen entsprechen. An der Y-Achse sind die prozentualen Anteile der gefundenen Werte relativ zur besten gefundenen Lösung der sechs Algorithmen jeder Sequenz.

[Erde, Mars, Erde, Mars, Venus, Erde, Venus, 2001 TW229], die beste Lösung aller Algorithmen 2.0586 wurde hierbei vom HJ gefunden und der zweitbeste Wert 0.8338 wurde vom Zufall gefunden. Dies entspricht damit einem prozentualen Anteil von $\frac{0.8338}{2.0586} = 0.405$ für den Zufall und $\frac{2.0586}{2.0586} = 1$ für HJ. Auffällig sind die Unterschiede zwischen den Algorithmen. Sehr oft hat der HJ den jeweils besten Wert gefunden, gefolgt vom Zufall. Der HC hat ein einziges mal die beste Lösung gefunden. Es liegt die Vermutung nahe, dass HC auf den ersten Blick, genau wie der SA, für diese Problemfamilie eher ungeeignet zu sein scheint. HJ scheint der geeignetste Algorithmus zu sein und DE und PSO bilden im Vergleich die mittlere Basis.

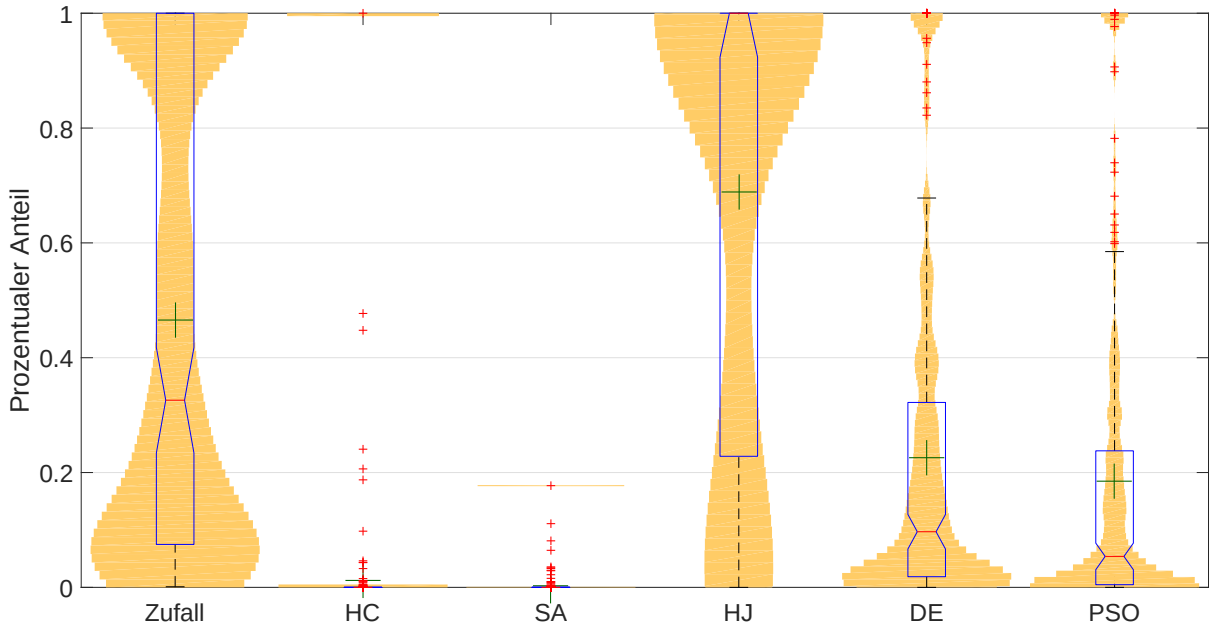


Abbildung 6.2: Boxplot der gemessenen Daten für das Ranking:
Es sind gemessene Daten für sechs Algorithmen, die auf der X-Achse zu sehen sind, dargestellt. Die Y-Achse entspricht den prozentualen Anteilen der gefundenen Werte relativ zur besten Lösung der sechs Algorithmen jeder Sequenz.

Um diese Vermutungen stärken zu können, wurde eine statistische Auswertung der Daten durchgeführt, deren Ergebnis in Abbildung 6.2 zu sehen ist. An der X-Achse sind die jeweiligen Algorithmen zu sehen und an der Y-Achse wieder die prozentualen Anteile bezüglich der besten gefundenen Lösung. Hierbei handelt es sich um übliche Boxplots: Innerhalb der Box liegen 50% der gemessenen Daten. Die Box wird durch den Median, zu sehen am Querstrich in der Box, geteilt, sodass in der Box jeweils ober- und unterhalb des Medians 25% der gemessenen Daten liegen. Der gekerbten Bereich (*Notches*) der Box entsprechen Konfidenzintervallen, das heißt, dass innerhalb dieser Begrenzungen mit 95%-iger Wahrscheinlichkeit der Median liegt. Werte

über und unter der Box sind jeweils 25% der gemessenen Daten. Sogenannte *milde* Ausreißer sind innerhalb der Whiskers (oder auch *Antennen* genannt) zu erkennen. Dabei liegen über und unter den jeweiligen Whisker 2,5% der gemessenen Daten. *Extreme* Ausreißer die in keinen der genannten Bereiche fallen, sind seperat als kleines Kreuz gekennzeichnet. Das große Kreuz entspricht dem Mittelwert der gemessenen Daten. Es wurde zusätzlich ein Violin-Plot unter den Box-Plot gelegt, der die allgemeine Verteilung der Daten veranschaulicht.

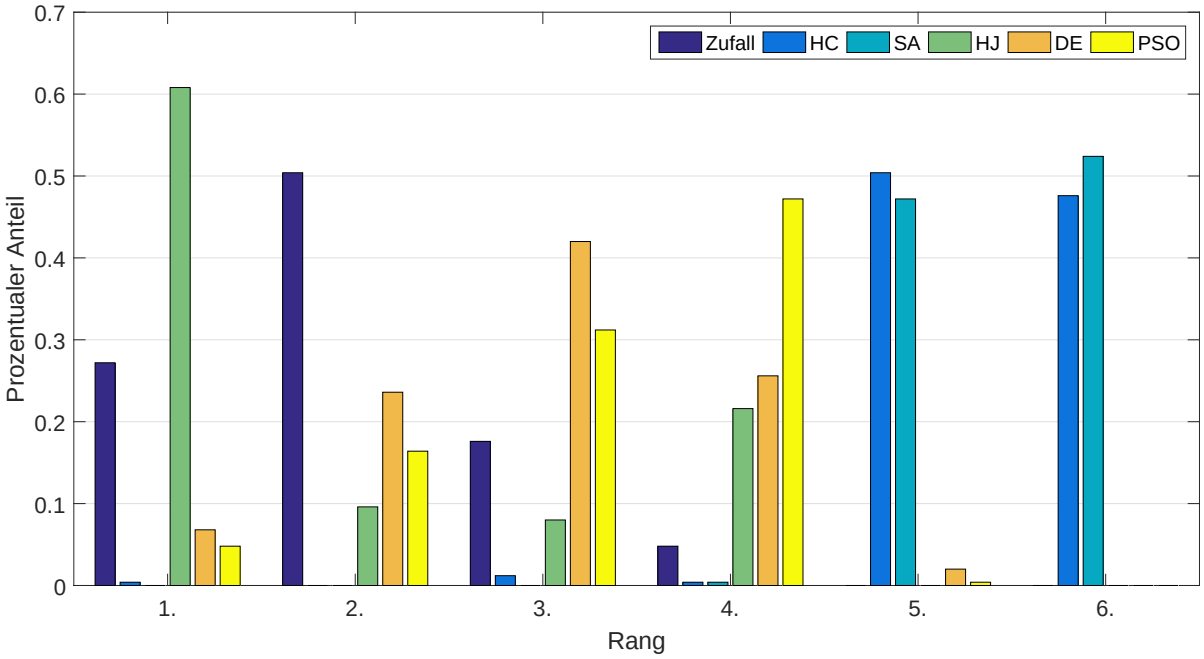


Abbildung 6.3: Ranking der gemessenen Daten:
Es sind auf der X-Achse sechs Ränge des Rankings erkennbar. Die Y-Achse entspricht den prozentualen Anteilen der gefundenen Werte relativ zur besten Lösung der sechs Algorithmen jeder Sequenz. Für jede dieser Ränge sind Balken für jeden Algorithmus dargestellt, die angeben, wie oft dieser Algorithmus im Bezug auf die gemessenen Daten in diesem Rang vertreten war.

Wie zuvor schon vermutet, zeigt sich in Abbildung 6.2 die Dominanz des HJ. Der Violin-Plot zeigt hierbei eine Ansammlung der Daten bei dem Wert 1, der Median liegt auch bei 1 und somit sind mindestens 50% der gemessenen Daten jeweils die beste Lösung gewesen. Der Zufall als Nachfolger zeigt auch ein überraschend gutes Ergebnis. Zwar liegt der Median im Vergleich *nur* bei ungefähr 0.33, doch findet der Zufall durchaus auch die beste Lösung, auch wenn 50% der Werte zwischen ungefähr zwischen 0.33 und 1 doch relativ weit streuen. Die prozentualen Anteile beider Algorithmen finden sich aber auch nahe von 0 wieder, was sich durch die gefundenen Lösungen anderer Algorithmen erklären lässt. Die Ergebnisse des DE und

PSO sehen ähnlich aus. Hier liegt der Median jeweils unter 0.1, wobei diese beiden Algorithmen auch mal die beste Lösung gefunden haben. Diese Lösungen sind aber als *extreme* Ausreißer anzusehen, da sie oberhalb des oberen Whiskers vorkommen, was heißt, dass sie in höchstens 2.5% der Fälle aufgetreten sind. Auch der HC hat einmal die beste Lösung gefunden, aber da der obere Whisker ungefähr bei dem Wert 0.02 liegt, scheint dieser einzelne Fund zufällig gewesen zu sein. Am schlechtesten schneidet der SA ab. Das Maximum liegt hier bei 0.18 und über dem Wert 0 sind auch nur *extreme* Ausreißer erkennbar.

Die zuvor betrachteten Algorithmen wurden für jedes konkrete Problem nach ihrer gefundenen Lösung sortiert, sodass insgesamt 250 Rankings entstanden sind. Das Ergebnis ist in Abbildung 6.3 zu sehen. An der X-Achse sind die jeweiligen Ränge 1 bis 6 zu sehen und an der Y-Achse wie oft, relativ zu den 250 Durchläufen, ein Algorithmus den jeweiligen Rank im Verhältnis erreicht hat. Der HJ dominiert den 1. Rang in ungefähr 61,4% der Fälle und ist insgesamt in den Rängen 1 bis 4 zu sehen, wobei HJ bei letzterem in ungefähr 22,1% der Fälle vorkommt. Der Zufall dominiert mit 60,5% den 2. Rang und ist ebenfalls in den Rängen 1 bis 4 erkennbar, wobei der Zufall in 28% der Fälle auch im 1. Rang vorkommt. DE hat seine höchste Anhäufung bei Rang 3 gefolgt von Rang 4 mit 26,2% und Rang 2 mit 23,2%. Die Daten des PSO fokussieren den 4. Rang mit 48,4% und fallen in Richtung des 1. Rangs weiter ab. Der HC und SA teilen sich ganz klar Rang 5 sowie Rang 6 und sind somit mit großem Abstand bezüglich dieses Ranking die schlechtesten Algorithmen für die Problemfamilie GTOC1.

6.3 Ergebnisse

In diesem Abschnitt werden die Optimierungsverfahren bezüglich des zuvor ermittelten Rankings evaluiert. Es wurden zuvor 250 konkrete Probleme untersucht, die jetzt in zwei Teile aufgespalten werden. Die ersten 125 konkreten Probleme dienen für die beiden Verfahren als Basis für die in Kapitel 5 dargestellten Regressionsmodelle. Nachdem die Regressionsmodelle mit Daten gefüllt wurden und so ein internes Modell entstanden ist, dienen die zweiten 125 konkreten Probleme als direkten Vergleich mit den zweiten 125 aus dem Ranking. Beide Verfahren werden dahingehend untersucht, wie sich die geschätzten Lösungen für die jeweiligen Algorithmen verteilen und die gut diese Schätzungen bezüglich des zuvor aufgestellten Rankings passen.

Als erstes wird DL betrachtet. In Abbildung 6.4 sind an der X-Achse die sechs Algorithmen zu sehen. An der Y-Achse sind ebenfalls prozentuale Anteile zu sehen, die hier aber relativ zur besten geschätzten Lösung sind. Der HJ dominiert beim DL sehr stark im Vergleich zu allen anderen Algorithmen. Die Box und die Whiskers beinhalten jeweils durchgehend den Wert 1, sodass HJ also in mindestens 95% der Fälle den besten geschätzten Wert besitzt oder einen der

sehr nah an ihm ist. Die restlichen maximal 5% der Fälle belaufen sich auf *extreme* Ausreißer. Der Zufall, DE und PSO wurden vom DL, wenn auch selten, auch als Algorithmus mit bester Lösung eingeschätzt, was die *extremen* Ausreißer bei HJ erklärt. Als zweitbester eingeschätzter Algorithmus wurde der Zufall bestimmt. Hierbei fokussiert sich die Box auf Werte zwischen 0.08 und 0.57, was einen schmaleren Rahmen darstellt als es in Abbildung 6.2 zu sehen war. Auch der Median ist schlechter geschätzt worden, als er gemessen wurde. DE und PSO geben ein ähnliches Bild ab, wobei der PSO schlechter geschätzt wurde. Bis auf *extreme* Ausreißer wurden diese Algorithmen nicht nennenswert gut eingeschätzt. HC und SA wurden mit großem Abstand also Algorithmen mit schlechtester geschätzter Lösung eingestuft. Die jeweiligen Maxima sind jeweils 0.02 beim HC und 0.015 beim SA.

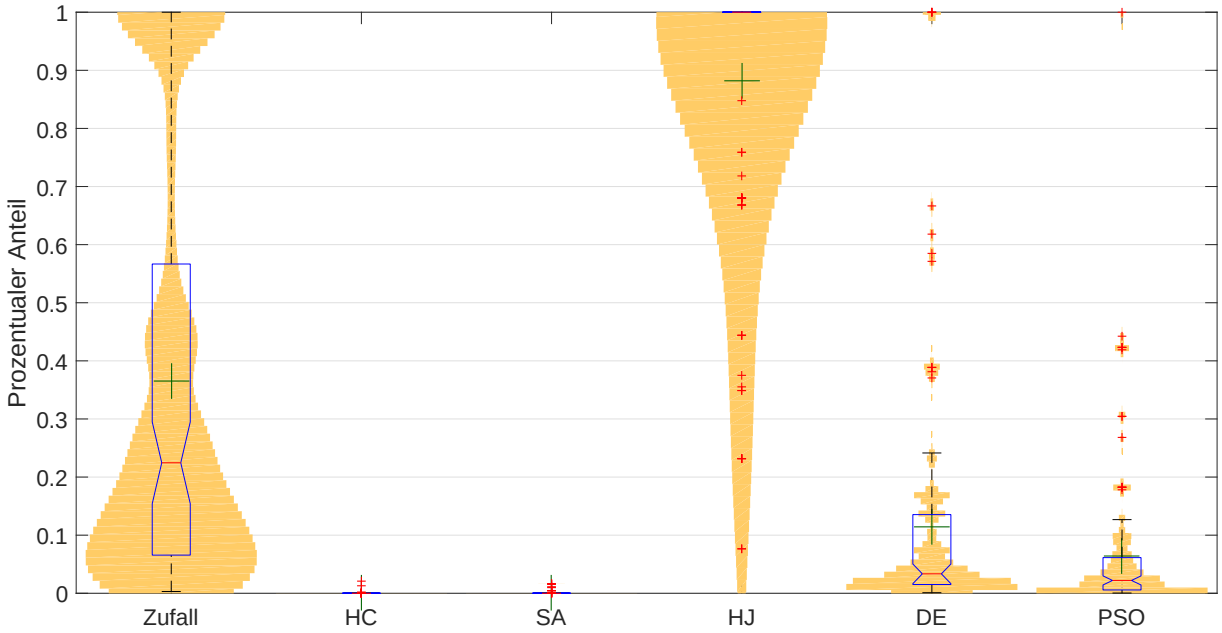


Abbildung 6.4: Boxplot der geschätzten Daten aus *Direct Learning*:
Es sind geschätzte Daten für sechs Algorithmen, die auf der X-Achse zu sehen sind, dargestellt. Die Y-Achse entspricht den prozentualen Anteilen der geschätzten Werte relativ zur besten geschätzten Lösung der sechs Algorithmen jeder Sequenz.

Es wurde nach Betrachtung der geschätzten Lösungen des DL ein Ranking bezüglich der geschätzten Lösungen erstellt, welches in Abbildung 6.5 zu sehen ist. Hier ergibt sich ein ähnliches Bild wie in Abbildung 6.3. Der HJ wurde hier in ungefähr 77,2% der Fälle auf den 1. Rang gesetzt und ist auch in den Rängen 2 bis 4 erkennbar. Der Zufall kommt mit 20,1% in Rang 1 vor, aber ist der Zufall in Rang 2 mit 56,7% am häufigsten vertreten und nimmt in den Rängen

3 und 4 ab. Der DE und der PSO verteilen sich auf den Rängen 3 und 4 mit jeweils 38,4% bis 51,4%, wobei der PSO in 18% der Fälle auch als der zweitbeste Algorithmus eingeschätzt wurde. Der HC ist mit 82,9% der Fälle auf Rang 5 und der SA in 81,2% der Fälle auf den letzten Rang eingestuft worden. Diese beiden Algorithmen kommen im Rang des anderen mit 14,1% bis 17% der Fälle vor, spielen in den höheren Rängen aber keine signifikante Rolle.

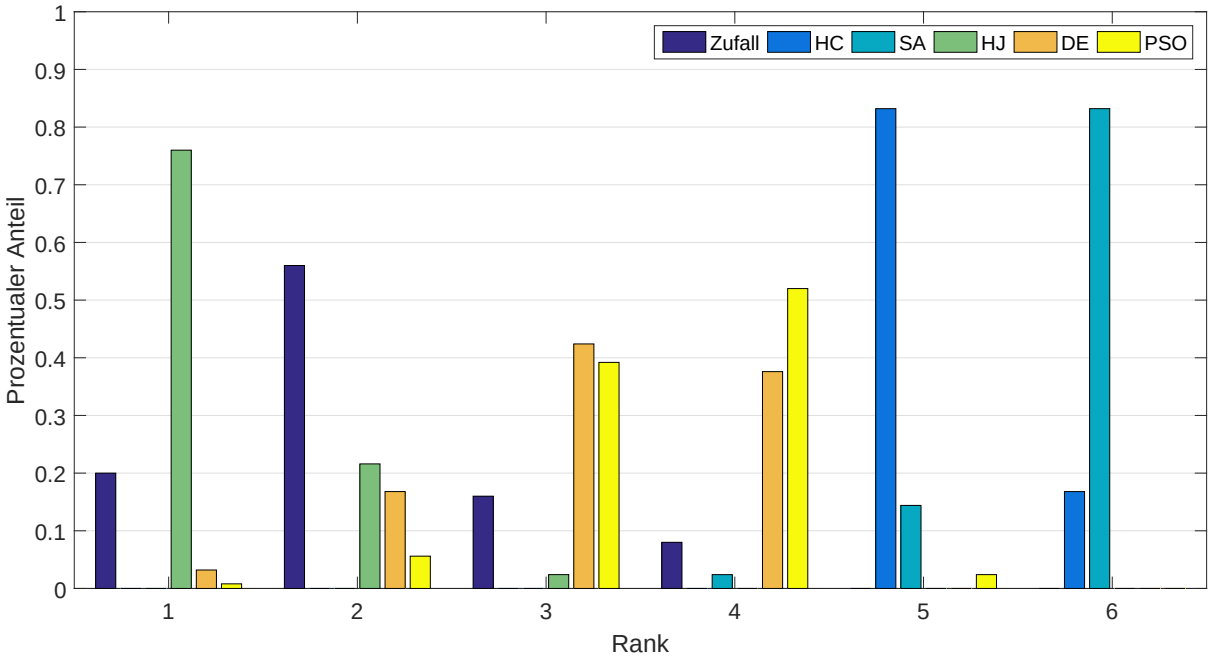


Abbildung 6.5: Ranking der geschätzten Daten aus *Direct Learning*:
Es sind auf der X-Achse sechs Ränge des Rankings erkennbar. Die Y-Achse entspricht den prozentualen Anteilen der geschätzten Werte relativ zur besten geschätzten Lösung der sechs Algorithmen jeder Sequenz. Für jede dieser Ränge sind Balken für jeden Algorithmus dargestellt, die angeben, wie oft dieser Algorithmus im Bezug auf die geschätzten Daten in diesem Rang vertreten war.

Es folgt die selbe Betrachtung wie beim DL jetzt für HINT. Für HINT wurde die Eigenschaftsvektorgröße $m = 4$ festgelegt. In Abbildung 6.6 sind die Box- mit Violin-Plot bezüglich der besten geschätzten Lösung zu sehen. Die Bedeutung der Achsen ist die gleiche wie in Abbildung 6.4. Auffällig ist, es keinen Algorithmus außer dem HJ gibt, der jemals mit einer besten Lösung eingeschätzt wurde und dass die Anzahl an *extremen* Ausreißern im Gegensatz zu DL stark abgenommen hat. Der Zufall hat mit HINT einen höheren Median 0.33 als beim DL und die Werte streuen weniger, aber es kommen durchaus Schätzungen nahe 0 vor. Die möglichen Lösungen des DE sind insgesamt höher geschätzt worden, was unter anderem am Median und an der Box erkennbar ist. Die Schätzungen des PSO bewegen sich um den Wert 0.06 herum

und sind mit einigen wenigen Ausreißern gegen 0 und 0.11 erkennbar. HC und SA sind wieder mit großem Abstand als am Schlechtesten einstuft worden. Alle geschätzten Werte grenzen bei den beiden an 0.

Nach der Betrachtung der Schätzungswerte wird wieder ein Ranking erstellt, welches in Abbildung 6.7 dargestellt ist. Wie zuvor schon betrachtet, ist es jetzt keine Überraschung, dass der HJ in 100% der Fälle Rang 1 besetzt. Rang 2 besetzt der Zufall in 100% der Fälle. DE und PSO kommen ausschließlich in den Rängen 3 und 4 vor, wobei DE häufiger Rang 3 besetzt. Das Verhältnis des Auftretens beider Algorithmen in Rang 3 und 4 ist 77:33. Fast das selbe Bild ergibt sich für HC und SA auf den Rängen 5 und 6. Hier ist, andersrum als beim DL, der SA öfter besser eingeschätzt worden als der HC mit dem Verhältnis 76:34.

Bis hier wurden DL und HINT im Kontext ihrer eigenen Einschätzungen betrachtet und es wurde jeweils ein Ranking aus den Schätzungen erstellt. Abschließend wird gegenübergestellt wie gut die Rankings auf Basis der geschätzten Werte zu dem Ranking der gemessenen Werte zusammenpassen.

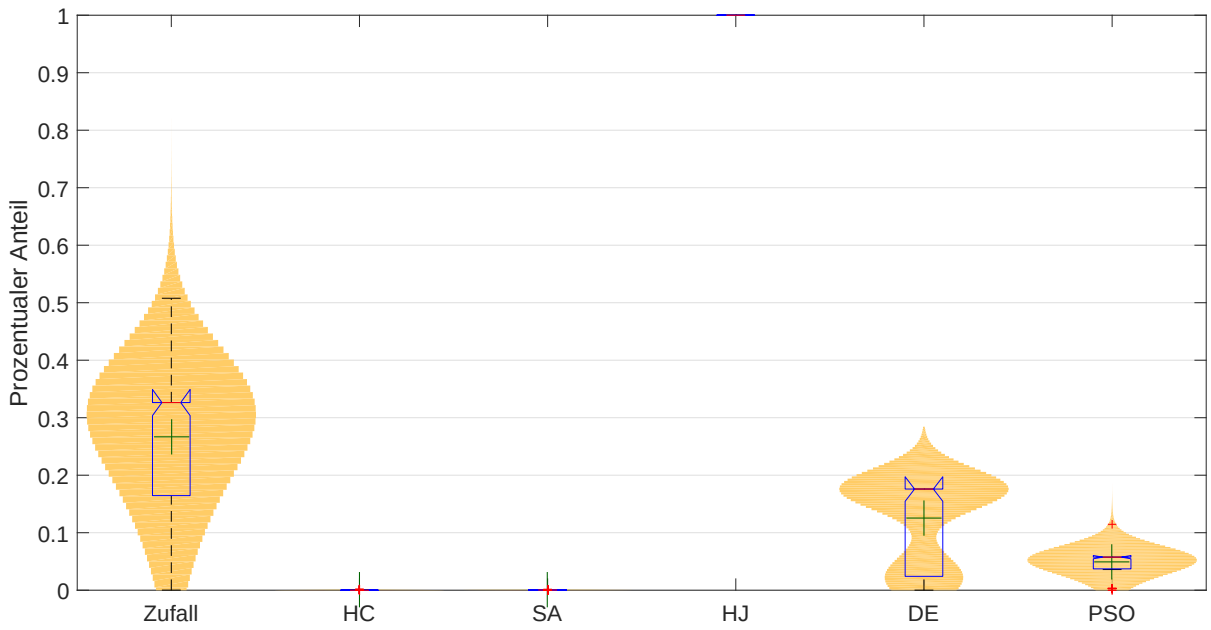


Abbildung 6.6: Boxplot der geschätzten Daten aus *Hint-Based Optimisation*:
Es sind geschätzte Daten für sechs Algorithmen, die auf der X-Achse zu sehen sind, dargestellt. Die Y-Achse entspricht den prozentualen Anteilen der geschätzten Werte relativ zur besten geschätzten Lösung der sechs Algorithmen jeder Sequenz.

In Abbildung 6.8 sind zwei Colormaps zu sehen. Die X-Achsen beschreiben jeweils den Rang der Algorithmen auf Basis der gemessenen Daten. Die Y-Achse beschreibt den geschätzten Rang mit DL beziehungsweise HINT. Die Farbe eines Kästchens entspricht der Trefferquote in Prozent mit der sich der geschätzte Rank mit dem gemessenen Rank deckt. Im Idealfall würden also auf der Hauptdiagonalen Trefferquoten von 100% zu sehen sein. Es zeigt sich, dass bezüglich beider Optimierungsverfahren die jeweiligen Trefferquoten Tendenzen zur Hauptdiagonalen haben. Allerdings streuen beide Verfahren, was jeweils in beiden Richtungen bei Rängen 1 bis 4 erkennbar ist, wobei die Streuung beim HINT geringer ist. Auch sind beide Verfahren im Verhältnis recht treffsicher mit 48,8% bis 52,8% was die Einschätzung der Ränge 5 und 6 betrifft. Die besten Trefferquoten sind, neben den eben erwähnten, bei DL an Stelle $(1, 1)$ mit 50,4% und bei HINT an den Stellen $(1, 1)$ mit 56% und $(2, 2)$ mit 49,6% zu finden.

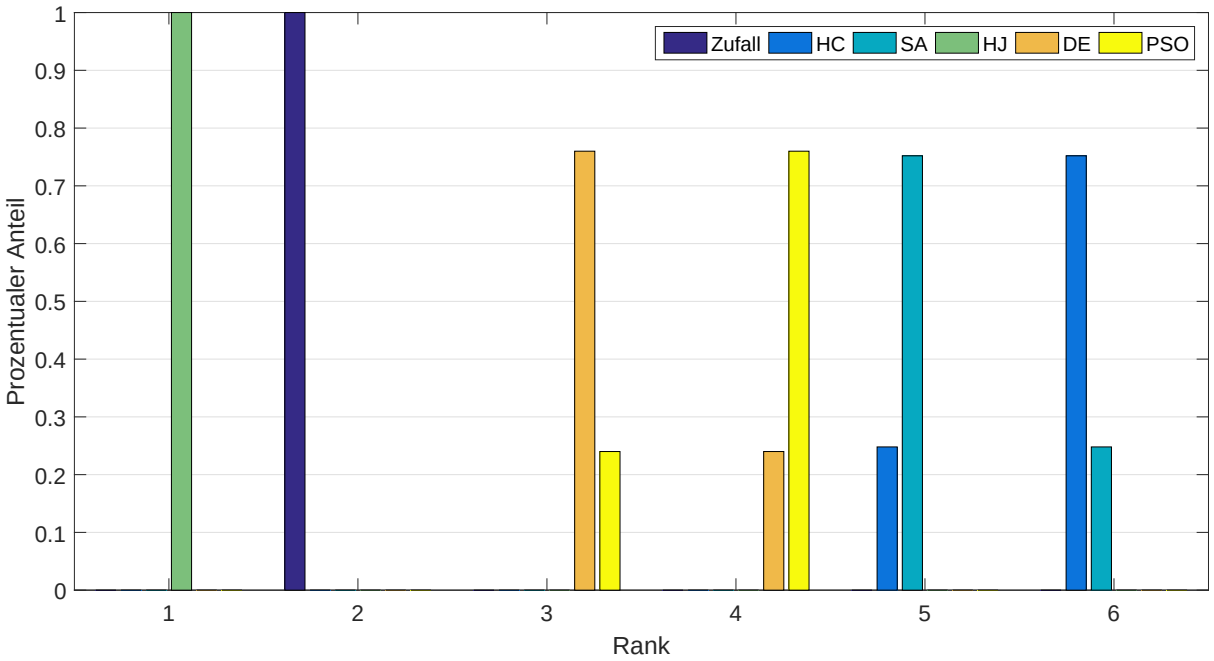


Abbildung 6.7: Ranking der geschätzten Daten aus *Hint-Based Optimisation*:
Es sind auf der X-Achse sechs Ränge des Rankings erkennbar. Die Y-Achse entspricht den prozentualen Anteilen der geschätzten Werte relativ zur besten geschätzten Lösung der sechs Algorithmen jeder Sequenz. Für jede dieser Ränge sind Balken für jeden Algorithmus dargestellt, die angeben, wie oft dieser Algorithmus im Bezug auf die geschätzten Daten in diesem Rang vertreten war.

Abschließend wurde eine Auswertung der Daten aus den Colormaps mit Hilfe der *mittleren quadratischen Abweichung* durchgeführt. Die geschätzten Ränge von DL und HINT werden

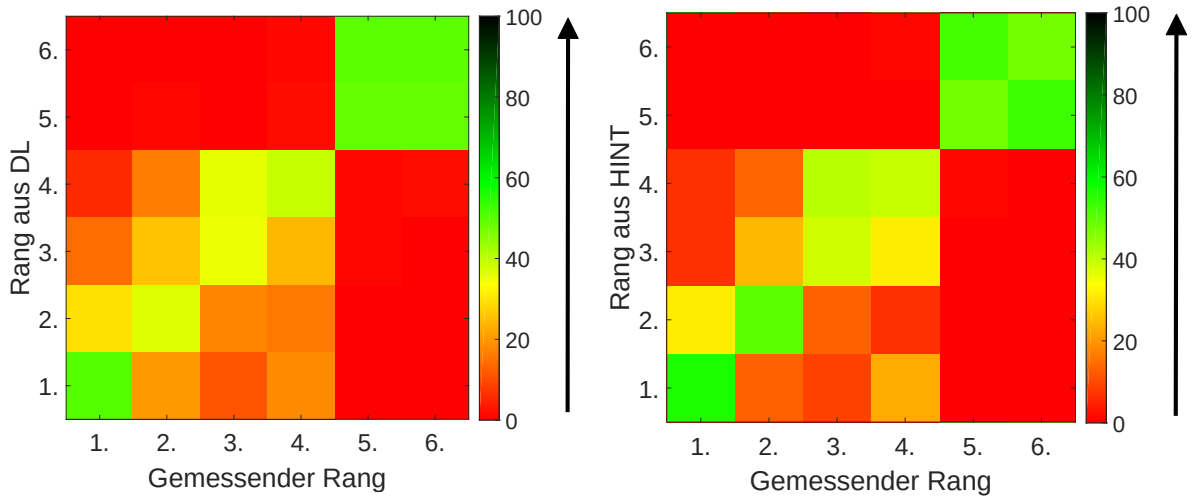


Abbildung 6.8: Vergleich Direct Learning und Hint-based Optimisation:
Es sind zwei Colormaps zu sehen. Die X-Achsen beschreiben jeweils den Rang der Algorithmen auf Basis der gemessenen Daten. Die Y-Achse beschreibt den geschätzten Rang mit DL beziehungsweise HINT. Die Farbe eines Kästchens entspricht der Trefferquote in Prozent mit der sich der geschätzte Rank mit dem gemessenen Rank deckt.

somit jeweils mit den gemessenen Rängen verrechnet. Die *mittlere quadratische Abweichung* beim DL ist 3.4773 und beim HINT 3.0107.

6.4 Diskussion

Beide Optimierungsverfahren DL und HINT haben laut gemessenem Ranking die besseren Algorithmen hauptsächlich gut eingeschätzt, darunter der HJ und Zufall. Dass der Zufall im Verhältnis so gut abschneidet, ist erstmal verwunderlich. Bei der Auswertung von Eigenschaftsvektoren ist es auch mal vorgekommen, dass deren Elemente (was im eigentlichen Sinne Steigungen entsprechen) gegen 0 gehen beziehungsweise der Eigenschaftsvektor gegen den Nullvektor geht. Dies ist ein Indiz dafür, dass innerhalb von $\mathcal{L}$ sogenannte *Plateaus*, an denen großflächig sehr geringe Steigungen vorkommen, existieren und Algorithmen ohne signifikant bessere Werte zu finden auf den Plateaus hängen bleiben. Dies kann erklären, dass der Zufall so gut abgeschnitten hat, da der Zufall beliebig springt und so nach der *Nadel im Heuhaufen* sucht ohne die Bewegung von der jeweiligen Funktion abhängig zu machen. Dies zeigt weiterhin, dass die Algorithmen HC und SA durch kurzsichtige Suche in der näheren Umgebung hier mit großem Abstand am Schlechtesten abschneiden und weitflächige Algorithmen von Vorteil sind. PSO

und DE sind in den Rankings am Häufigsten in den Rängen 3 und 4 vorgekommen. Durch den Einfluss der einzelnen Partikel aus den Populationen, dem Anteil des Zufalls beim PSO und dem Crossover beim DE konnten PSO und DE bessere Werte finden, da sie so einen größeren Raum abdecken als HC und SA. HJ als (bis auf die zufälligen Startposition) deterministischer Algorithmus deckt mit seinem strukturierten Vorgehen und seinen $2n$ Distanzmessungen ebenfalls einen großen Raum ab. Hat der HJ eine *Nadel im Heuhaufen* gefunden, konvergiert der HJ durch die Halbierung seiner Distanzen schnell. Dies sind alles Faktoren, die für das gute Abschneiden des HJ bei dieser Problemfamilie verantwortlich sind.

Jetzt zum Vergleich der Optimierungsverfahren DL und HINT. Beide Verfahren haben im Bezug auf das gemessene Ranking die geeignetsten Algorithmen gefunden und die Ränge, mit Ausnahme von Rang 5 und 6 bei HINT, richtig zugeordnet. Der Aufwand zusätzlicher Regressionsmodelle für die Eigenschaften bei HINT hat sich in der Hinsicht gelohnt, da weniger Streuung und *extreme* Ausreißer in den Schätzungen der Lösungen vorkamen als bei DL. Im direkten Vergleich war in den Colormaps zu erkennen, dass die Optimierungsverfahren ähnlich gut im Bezug auf das gemessene Ranking geschätzt haben. Aber auch hier sind im Detail Unterschiede zu erkennen. HINT hat eine geringere Streuung gegenüber der Hauptdiagonalen und bessere Trefferquoten auf der Hauptdiagonalen. Auch im Vergleich der *mittleren quadratischen Abweichung* schneidet HINT besser ab als DL. Insgesamt ist HINT für die Problemfamilie GTOC1 also das bessere Optimierungsverfahren von beiden.

7 Fazit

Diese Arbeit beschäftigte sich mit der Evaluierung des Optimierungsverfahrens *Hint-Based Optimisation* im Kontext der Berechnung interplanetarer Bahnkurven. Es wurde eine Problemfamilie aus dem von der *European Space Agency* initialisierten Wettbewerb *Global Trajectory Design Competition* für die Untersuchung ausgewählt.

Nach einer Einführung in die Himmelsmechanik und deren Methoden wurde die Problemfamilie *Save The Earth* aus dem 1. Durchlauf der *Global Trajectory Design Competition* vorgestellt. Das Ziel war diese Problemfamilie mittels des Optimierungsverfahrens *Hint-Based Optimisation* auf einen passenden Optimierungsalgorithmus bezüglich der Qualität der gefundenen Lösung automatisiert zu untersuchen. Diesem Verfahren wurde ein simpleres Verfahren *Direct Learning* gegenübergestellt, um festzustellen in wie fern sich der Mehraufwand von *Hint-Based Optimisation* in der Qualität der gefundenen Lösung widerspiegelt.

Der *Hooke-Jeeves-Algorithmus* und der Zufall fanden für diese Problemfamilie oft beste Lösungen, gefolgt von den Optimierungsalgorithmen *Differential Evolution* und *Particle Swarm Optimization*. *Hill Climber* und *Simulated Annealing* scheinen ungeeignet bezüglich dieser Problemfamilie zu sein. Durch die Betrachtung der Eigenschaften aus *Hint-Based Optimisation* zeigte sich, dass diese Problemfamilie große und/oder viele *Plateaus* besitzt, was diese Problemfamilie zur Suche nach der *Nadel im Heuhaufen* macht. Diese Eigenschaft trägt dazu bei, dass stark umgebungsorientierte Optimierungsalgorithmen wie *Hill Climber* und *Simulated Annealing* selten gute Lösungen fanden, wobei der Zufall durch seine Streuung und der *Hooke-Jeeves-Algorithmus* durch seine hohe Raumabdeckung öfter gute oder sogar beste Lösungen fanden.

Als Basis wurde ein Ranking aus Messpunkten erstellt, das die untersuchten Algorithmen hinsichtlich ihrer gefundenen Lösung sortiert. Es hat sich gezeigt, dass sowohl *Direct Learning* also auch *Hint-Based Optimisation* sehr gut für die Auswahl der Algorithmen funktionieren, wie es ein Ranking bezüglich der Schätzwerte gezeigt hat, welches ähnliche Muster wie die gemessene Variante zeigt. Das gemessene Ranking wurde mit den Schätzwerten von *Direct Learning* sowie *Hint-Based Optimisation* verglichen und es war ersichtlich, dass *Hint-Based Optimisation* bezogen auf die Messdaten in der korrekten Prädiktion eines Rankings besser ist als *Direct Learning*.

8 Ausblick

In der Diskussion der Evaluation sind Plateaus als Problem für bestimmte Optimierungsalgorithmen, insbesondere umgebungsorientierte wie *Hill Climber* oder *Simulated Annealing*, erkannt worden. Da der Zufall durch die Suche nach der *Nadel im Heuhaufen* bei dieser Problemfamilie gut abschneidet, wären Strategien denkbar, die detektieren können, ob sich ein Messpunkt auf einem solchen *Plateau* befindet und entsprechend zufällig eine neue Position erhält um das aktuelle Plateau zu verlassen. Im Stand der Technik wurde der Ansatz des *Gravity Assist Space Pruning* angesprochen. Dieser Ansatz der Suchraumreduktion könnte unter Umständen die Größe von Plateaus einschränken, sodass Optimierungsalgorithmen, die bei dieser Problemfamilie schlecht abgeschnitten haben, dann vielleicht eher bessere Lösungen finden.

Ein anderer interessanter Aspekt wäre zu prüfen, ob die Laufzeit bezüglich der Bestimmung von Eigenschaften für den Eigenschaftsvektor und den damit verbundenen Regressionen im Verhältnis zum Qualitätsgewinn der gefunden Lösung steht. Des weiteren wäre die Betrachtung verschiedener Anzahlen von Eigenschaften beziehungsweise der unterschiedlicher Eigenschaftsvektorgröße m interessant und welche Vor- und Nachteile sich dadurch ergeben.

Desweiteren wäre zu prüfen, ob die unterschiedlichen Problemfamilien aus der *Global Trajectory Optimization Competition* oder die Änderung der erlaubten Raumschiff-Manöver von *Multiple Gravity Assist* auf *Multiple Gravity Assist One Deep Space* ähnliches Verhalten beim *Hint-Based Optimisation* aufzeigen.

Literatur

[1] Horst, Reiner ; Pardalos, Panos M.: *Handbook of global optimization*. Bd. 2. Springer Science & Business Media, 2013

[2] Rice, John R.: The algorithm selection problem. In: *Advances in computers* 15 (1976), S. 65–118

[3] Wolpert, David H. ; Macready, William G.: No free lunch theorems for optimization. In: *IEEE transactions on evolutionary computation* 1 (1997), Nr. 1, S. 67–82

[4] Wolpert, David H. ; Macready, William G.: Coevolutionary free lunches. In: *IEEE Transactions on Evolutionary Computation* 9 (2005), Nr. 6, S. 721–735

[5] John, George H. ; Kohavi, Ron ; Pfleger, Karl u. a.: Irrelevant features and the subset selection problem. In: *Machine learning: proceedings of the eleventh international conference*, 1994, S. 121–129

[6] Vinkó, Tamás ; Izzo, Dario ; Bombardelli, Claudio: Benchmarking different global optimisation techniques for preliminary space trajectory design. In: *58th International Astronautical Congress, International Astronautical Federation (IAF)*, 2007

[7] Jones, Terry: *Evolutionary algorithms, fitness landscapes and search*, Citeseer, Diss., 1995

[8] Izzo, Dario: *Institutions*. Website, 2016. – `http://sophia.estec.esa.int/gtoc_portal/` abgerufen am 21. Juni 2016

[9] He, Jun ; Reeves, Colin ; Witt, Carsten ; Yao, Xin: A note on problem difficulty measures in black-box optimization: Classification, realizations and predictability. In: *Evolutionary Computation* 15 (2007), Nr. 4, S. 435–443

[10] Schwefel, Hans-Paul: *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Bd. 1. Birkhäuser, Basel Switzerland, 1977

[11] Gelatt, CD ; Vecchi, MP u. a.: Optimization by simulated annealing. In: *Science* 220 (1983), Nr. 4598, S. 671–680

[12] Hooke, Robert ; Jeeves, Terry A.: "Direct Search"Solution of Numerical and Statistical Problems. In: *Journal of the ACM (JACM)* 8 (1961), Nr. 2, S. 212–229

[13] Kennedy, J.: Particle Swarm Optimization. In: *Neural Networks* Bd. 4. IEEE International Conference, 1995

[14] Price, Kenneth ; Storn, Rainer M. ; Lampinen, Jouni A.: *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006

[15] Izzo, Dario: Global optimization and space pruning for spacecraft trajectory design. In: *Spacecraft Trajectory Optimization* 1 (2010), S. 178–200

[16] Aarseth, Sverre J. ; Aarseth, Sverre J.: *Gravitational N-body simulations: tools and algorithms*. Cambridge University Press, 2003

[17] Montenbruck, Oliver: *Grundlagen der Ephemeridenrechnung*. Springer-Verlag, 2009

[18] Barrar, RB: An analytic proof that the Hohmann-type transfer is the true minimum two-impulse transfer / DTIC Document. 1962. – Forschungsbericht

[19] Shen, Haijun ; Tsiotras, Panagiotis: Optimal two-impulse rendezvous using multiple-revolution Lambert solutions. In: *Journal of Guidance, Control, and Dynamics* 26 (2003), Nr. 1, S. 50–61

[20] Jordan, JF: The Application of Lambert's Theorem to the Solution of Interplanetary Transfer Problems. In: *Jet Propulsion Laboratory* (1964), S. 32–521

[21] Broucke, RA: Gravity Assist. (1988)

[22] Vasile, Massimiliano ; Pascale, P D.: Preliminary design of multiple gravity-assist trajectories. In: *Journal of Spacecraft and Rockets* 43 (2006), Nr. 4, S. 794–805

[23] Labunsky, Alexei V. ; Papkov, Oleg V. ; Sukhanov, Konstantin G.: *Multiple gravity assist interplanetary trajectories*. CRC Press, 1998

[24] Izzo, Dario: *Institutions*. Website, 2015. – `http://sophia.estec.esa.int/gtoc_portal/?page_id=405` abgerufen am 21. Juni 2016

[25] Izzo, Dario: *GTOC9*. Website, 2016. – `http://sophia.estec.esa.int/gtoc_portal/?page_id=782` abgerufen am 21. Juni 2016

[26] Izzo, Dario: *Problem Description for the 1st ACT Competition on Global Trajectory Optimisation*. European Space Agency, 2005

[27] Leyton-Brown, Kevin ; Nudelman, Eugene ; Andrew, Galen ; McFadden, Jim ; Shoham, Yoav: A portfolio approach to algorithm selection. In: *IJCAI* Bd. 1543, 2003, S. 2003

[28] Cover, Thomas ; Hart, Peter: Nearest neighbor pattern classification. In: *IEEE transactions on information theory* 13 (1967), Nr. 1, S. 21–27

[29] Kadioglu, Serdar ; Malitsky, Yuri ; Sabharwal, Ashish ; Samulowitz, Horst ; Sellmann, Meinolf: Algorithm selection and scheduling. In: *International Conference on Principles and Practice of Constraint Programming* Springer, 2011, S. 454–469

[30] Brazdil, Pavel B. ; Soares, Carlos: A comparison of ranking methods for classification algorithm selection. In: *European Conference on Machine Learning* Springer, 2000, S. 63–75

[31] Kohavi, Ron u. a.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *Ijcai* Bd. 14, 1995, S. 1137–1145

[32] Puterman, Martin: *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994

[33] Watkins, Christopher J. ; Dayan, Peter: Q-learning. In: *Machine learning* 8 (1992), Nr. 3-4, S. 279–292

[34] Lagoudakis, Michail G. ; Littman, Michael L.: Algorithm Selection using Reinforcement Learning. In: *ICML* Citeseer, 2000, S. 511–518

[35] Sharaf, MA ; Saad, AN ; Nouh, MI: Lambert universal variable algorithm. In: *Arabian Journal for Science and Engineering* 28 (2003), Nr. 1, S. 87–98

[36] Householder, Alston S.: *The Numerical Treatment of a Single Nonlinear Equation*. McGraw Hill, 1970

[37] Izzo, Dario: Revisiting Lambert's problem. In: *Celestial Mechanics and Dynamical Astronomy* 121 (2015), Nr. 1, S. 1–15

[38] Zhang, Ying ; Yue, Xiaokui ; He, Liang: Design of multiple gravity assist trajectories with deep space maneuver. In: *Chinese Space Science and Technology* 33 (2013), Nr. 2, S. 61–66

[39] Izzo, Dario: 1st ACT global trajectory optimisation competition: Problem description and summary of the results. In: *Acta Astronautica* 61 (2007), Nr. 9, S. 731–734

[40] Petropoulos, Anastassios E. ; Longuski, James M.: Shape-based algorithm for the automated design of low-thrust, gravity assist trajectories. In: *Journal of Spacecraft and Rockets* 41 (2004), Nr. 5, S. 787–796

[41] Sims, Jon A. ; Flanagan, Steve N.: *Preliminary design of low-thrust interplanetary missions*. 1997

[42] Petropoulos, Anastassios E. ; Kowalkowski, Theresa D. ; Vavrina, Matthew A. ; Parcher, Daniel W. ; Finlayson, Paul A. ; Whiffen, Gregory J. ; Sims, Jon A.: 1st ACT global trajectory optimisation competition: Results found at the Jet Propulsion Laboratory. In: *Acta Astronautica* 61 (2007), Nr. 9, S. 806–815

[43] Damme, Carlos C.: *MITRADES - Interactive Trajectory Design with MATLAB*. 2006

[44] Davis, Lawrence: *Handbook of genetic algorithms*. Van Nostrand Reinhold, 1991

[45] Damme, Carlos C. ; Prieto-Llanos, Tomás ; Cadenas, Raúl ; Gil-Fernández, Jesús ; Graziano, Mariella: 1st ACT global trajectory optimization competition: Results found at GMV. In: *Acta Astronautica* 61 (2007), Nr. 9, S. 786–793

[46] Cano, JL ; Bello, M ; Rodriguez-Canabal, J: Navigation and Guidance for Low-Thrust Trajectories, LOTNAV. In: *18th International Symposium on Space Flight Dynamics* Bd. 548, 2004, S. 609

[47] Mora, Miguel B. ; Cano, Juan L.: 1st ACT global trajectory optimisation competition: Results found at DEIMOS Space. In: *Acta Astronautica* 61 (2007), Nr. 9, S. 794–805

[48] Nelder, John A. ; Mead, Roger: A simplex method for function minimization. In: *The computer journal* 7 (1965), Nr. 4, S. 308–313

[49] Bertrand, R ; Ceolin, T ; Epenoy, R: First ACT global trajectory optimisation competition results found at CNES/CS. In: *Acta Astronautica* 61 (2007), Nr. 9, S. 763–768

[50] Ingber, Lester: Very fast simulated re-annealing. In: *Mathematical and computer modelling* 12 (1989), Nr. 8, S. 967–973

[51] Csendes, Tibor: Nonlinear Parameter Estimation by Global Opitmization - Efficiency and reliability. In: *Acta Cybern.* 8 (1988), Nr. 4, S. 361–372

[52] Vinkó, T ; Izzo, D ; Pinna, F: Learning the best combination of solvers in a distributed global optimization environment. In: *Proceedings of Advances in Global Optimization: Methods and Applications (AGO), Mykonos, Greece* (2007), S. 13–17

[53] Izzo, Dario ; Becerra, Victor M. ; Myatt, Darren R. ; Nasuto, Slawomir J. ; Bishop, J M.: Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories. In: *Journal of Global Optimization* 38 (2007), Nr. 2, S. 283–296

[54] Price, Kenneth V.: Differential evolution. In: *Handbook of Optimization*. Springer, 2013, S. 187–214

[55] Zambrano-Bigiarini, Mauricio ; Clerc, Maurice ; Rojas, Rodrigo: Standard particle swarm optimisation 2011 at cec-2013: A baseline for future pso improvements. In: *2013 IEEE Congress on Evolutionary Computation* IEEE, 2013, S. 2337–2344