Technische Universität Darmstadt

Fachbereich Informatik
Fachgebiet Simulation und Systemoptimierung
Prof Dr. Oskar von Stryk

Diplomarbeit

Physikalisch motivierte Handlungsreisendenprobleme mit differenzierbaren Wegen - Bestimmung von Startschätzungen und Schranken

Physically motivated Traveling Salesman Problems on smooth curves - Finding initial estimates and bounds

Bearbeitet von:	Simon Erassmy
Matrikelnummer:	1077922
Studiengang:	Wirtschaftinformatik

Aufgabensteller:	Prof. Oskar von Stryk
Betreuer:	Dipl.-Math. Christian Reinl
Abgabetermin:	17.11.2006

Zusammenfassung

Diese Arbeit beschreibt die Entwicklung und Implementierung mehrerer Verfahren zur Generierung von Schranken und Startschätzungen für ein motorisiertes Handlungsreisendenproblem (engl. Motorized Traveling Salesman Problem - MTSP). Im Gegensatz zu einem herkömmlichen, graphentheoretischen Handlungsreisendenproblem (engl. Traveling Salesman Problem - TSP) bewegt sich hier ein physikalisch modelliertes Fahrzeug auf glatten Kurven durch die (x, y)-Ebene. Die Optimierung des Problems gestaltet sich schwieriger als bei einem klassischen TSP, da die Reihenfolge der Städte und die Reisekosten zwischen den Städten nicht mehr unabhängig voneinander sind. In einem ersten Ansatz werden verschiedene Splines zur Approximation der optimalen Kurven verwendet. Dabei wird untersucht, wie gut die Spline-Kurven ein physikalisch fundiertes Optimalsteuerungsproblem in Abhängigkeit von dem zugrunde liegenden Modell approximieren können. In einem weiteren Ansatz wird die Physik direkt bei der Konstruktion der Splines berücksichtigt.

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe.

Darmstadt, den 17.11.2006

Simon Erassmy

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Handlungsreisenden- oder auch Traveling-Salesman-Probleme (TSP) gehören zu den bekanntesten kombinatorischen Optimierungsproblemen. Ein praktisches Problem dieser Klasse könnte lauten: Ein Handlungsreisender möchte eine Anzahl von Kunden an verschiedenen Orten besuchen. Nach Abschluss der Besuche möchte er an seinen Ausgangsort zurückkehren. Welchen Weg soll er wählen (in welcher Reihenfolge soll er die Kunden besuchen), damit die dabei insgesamt zurückzulegende Entfernung so gering wie möglich ist?

Eine bedeutende Eigenschaft dieser Probleme ist, dass die Reisekosten für jedes Städtepaar fix und somit unabhängig von der restlichen Rundreise sind. Diese Eigenschaft ist nicht mehr erfüllt, wenn die Städte Punkte in der (x, y)-Ebene sind und der Handlungsreisende sich in einer glatten Kurve und ohne anzuhalten durch die Städte bewegen soll. Diese Problem wird auch als *motorisiertes Handlungsreisendenproblem* oder *Motorized Traveling Salesman Problem* (MTSP) bezeichnet [vSG00].

Besondere praktische Bedeutung hat diese Problemstellung z.B. für ein Löschflugzeug, das von einer Basis aus operiert und eine Reihe von Brandherden mit optimaler Flugbahn anfliegen soll. Die Flugbahn ist dabei eine glatte Kurve, die von den individuellen, flugmechanischen Eigenschaften des Flugszeugs bestimmt wird. Was unter einer „optimalen Flugbahn“ verstanden wird, ist Definitionssache. Eine mögliche Zielsetzung ist die Minimierung der Flugzeit, eine andere die Minimierung des Treibstoffverbrauchs. Ebenso kann eine gewichtete Kombination aus mehreren Zielsetzungen verwendet werden [Glo05].

Probleme, wie das eben beschriebene, gehören zur Klasse der *diskret-kontinuierlichen Optimalsteuerungsprobleme*. Diese werden durch Anwendung eines Transformationsverfahrens wie der *direkten Kollokation* zu *nichtlinearen gemischt-ganzzahligen Optimierungsproblemen* oder *Mixed-Integer NonLinear Programs* (MINLP). Die noch recht junge Klasse von Optimierungsproblemen beinhaltet sowohl kontinuierliche als auch diskrete Variablen und kann sowohl eine nichtlineare Zielfunktion als auch nichtlineare Nebenbedingungen enthalten [BP03].

Da sowohl die kombinatorischen (*Mixed-Integer Program* - MIP) als auch die nichtlinearen Optimierungsprobleme (*NonLinear Program* - NLP) zu einer Klasse von theoretisch schwierigen Problemen (*NP*-vollständig) gehören, verwundert es nicht, dass MINLPs besonders schwer zu lösen sind.

NP ist eine Komplexitätsklasse, in der sich Entscheidungsprobleme befinden, die von einer nichtdeterministischen Turingmaschine in polynomialer Zeit entschieden werden können [Sch01]. Man bezeichnet ein Problem P als *NP-hart*, wenn sich alle Probleme der Klasse NP in polynomialer Zeit auf P reduzieren lassen. Ein Problem wird als NP-vollständig bezeichnet, wenn es NP-hart ist und in der Klasse NP liegt.

In der Praxis bedeutet dies, dass keine Algorithmen existieren, die NP-harte oder NP-vollständige Probleme in polynomialer Rechenzeit lösen können. Eine direkte Folge ist, dass solche Probleme nur für sehr kleine Problemgrößen in annehmbarer Zeit exakt gelöst werden können.

MINLPs lassen sich unter bestimmten Bedingungen in ein MIP und mehrere NLPs zerlegen. Zur exakten Lösung kommt dann eine Kombination von Techniken zur Lösung von kombinatorischen und nichtlinearen Optimierungsproblemen zum Einsatz. Für die Lösung des kombinatorischen Teilproblems werden dabei oft *Branch & Bound* Verfahren benutzt, für die Lösung der nichtlinearen Teilprobleme wird häufig *sequentielle quadratische Programmierung* (SQP) verwendet.

Die Forschung ist in diesem Bereich allerdings noch am Anfang ihrer Entwicklung, es wird ständig nach besseren Lösungsverfahren gesucht. Ziel dieser Arbeit ist es daher, die Lösungsverfahren für MTSPs zu verbessern. Bevor im folgenden Kapitel die aktuellen Lösungsverfahren, ihre Problematiken und die Zielsetzungen dieser Arbeit beschrieben werden, folgt in dieser Einleitung zunächst eine genaue Definition von Handlungsreisendenproblemen.

1.1 Klassische Handlungsreisendenprobleme

Klassische Handlungsreisendenprobleme werden im Allgemeinen als *graphentheoretische* Probleme formuliert. Die zu besuchenden Orte sowie der Ursprung sind die Knoten eines Graphen $G = (V, E, c)$. Dabei bezeichnet V die Menge aller Knoten und E die Menge aller Kanten. Die Kostenmatrix c der Dimension $V \times V$ enthält die Bewertungen aller Kanten $(i, j) \in E$. Die Funktion $c(i, j)$ liefert die Kosten der Strecke von Knoten i nach Knoten j.

Gesucht ist ein kostenminimaler Weg in G, der jeden Knoten aus V genau einmal enthält. Gilt $c(i, j) = c(j, i) \; \forall \; (i, j) \in E$, so spricht man von einem *symmetrischen* TSP, ansonsten von einem *asymmetrischen* TSP. Formal ist das asymmetrische TSP in der Graphentheorie wie folgt definiert [Dom97]:

Definition 1.1. Asymmetrisches TSP (graphentheoretisch)

Sei $G = (V, E, c)$ ein bewerteter Digraph mit n Knoten. Ein Zyklus ρ, der jeden Knoten von G genau einmal enthält, wird als **Rundreise** oder **Hamiltonscher**

Zyklus von G bezeichnet. Einen Zyklus ζ, der weniger als n Knoten enthält, wird als **Kurzzyklus** (engl. Subtour) von G bezeichnet. Das Problem der Bestimmung einer kürzesten Rundreise in einem gerichteten Graphen wird als **asymmetrisches TSP** bezeichnet.

Die Definition eines symmetrischen TSP erfolgt analog. Im Folgenden wird unter einem TSP immer das allgemeinere asymmetrische TSP verstanden, da sich jedes symmetrische TSP in ein asymmetrisches transformieren lässt.

Als nächstes wird die mathematische Modellierung von Handlungsreisendenproblemen vorgestellt. Neben den bereits beschriebenen Komponenten werden die Variablen x_{ij} $(i, j = 1, ... n)$ mit der folgenden Bedeutung verwendet:

$$x_{ij} = \begin{cases} 1 & \text{falls die Rundreise von Knoten } i \text{ unmittelbar nach Knoten } j \text{ führt} \\ 0 & \text{sonst} \end{cases}$$

Die Zielfunktion lautet:

$$\text{Minimiere } F(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{1.1}$$

unter den Nebenbedingungen

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{für } i = 1, ..., n \tag{1.2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \text{für } j = 1, ..., n \tag{1.3}$$

$$x_{ij} \in \{0, 1\} \qquad \text{für } i, j = 1, ..., n \tag{1.4}$$

$$\text{Bedingungen zur Verhinderung von Kurzzyklen} \tag{1.5}$$

Die Zielfunktion (1.1) ist bei dieser Formulierung die Summe der Bewertungen aller Kanten, die in der Lösung enthalten sind. Die Nebenbedingungen (1.2) sowie (1.3) stellen die notwendigen Bedingungen für eine Rundreise dar, nämlich dass jeder Knoten genau einmal erreicht und genau einmal wieder verlassen wird. Zusammen entsprechen diese Gleichungen (1.1) - (1.4) der Formulierung des, in polynomialer Zeit lösbaren, *linearen Zuordnungproblems* [Dom95]. Erst die, bisher noch nicht genauer spezifizierten, Bedingungen zur Verhinderung von Kurzzyklen machen Handlungsreisenden Probleme NP-hart.

Es gibt verschiedenen Möglichkeiten, Zyklusbedingungen zu formulieren. Die drei in [Dom97] vorgestellten Formulierungen unterscheiden sich sehr stark in der Anzahl der erforderlichen Nebenbedingungen. Aus Gründen der Übersichtlichkeit wird hier auf eine Darstellung aller Möglichkeiten verzichtet und hier mit den *Miller-Tucker-Zemlin-Bedingungen* nur die effizienteste Variante vorgestellt.

Für jeden Knoten $i \neq 1$ wird dabei eine reelle Hilfsvariable π_i eingeführt. Die Anzahl der Zyklusbedingungen beträgt für diese Variante $(n-1) \cdot (n-2)$. Sie lauten:

$$\pi_i - \pi_j + n x_{ij} \leq n - 1 \qquad \text{für alle } i, j = 2, ..., n \text{ mit } i \neq j \tag{1.6}$$

Die Bedingungen schließen sämtliche Zyklen, die Knoten 1 nicht enthalten aus. So verbleiben nur noch Zyklen, die zulässige Rundreisen darstellen.

1.2 Motorisierte Handlungsreisendenprobleme

Das motorisierte Handlungsreisendenproblem ist, im Gegensatz zum klassischen, nicht als graphentheoretisches Problem formulierbar, da die einzelnen Segmente des Weges nicht mehr unabhängig voneinander sind. Der „Handlungsreisende“ ist hier ein Fahrzeug, welches sich in der (x, y)-Ebene bewegt und in einer glatten Kurve alle Orte durchfahren soll.

Das Fahrzeug wird dabei in der einfachsten Version durch eine Punktmasse repräsentiert, die sich frei in der (x, y)-Ebene bewegen kann.Der Zustand des Fahrzeugs zum Zeitpunkt t entspricht den (x, y) Koordinaten, an denen es sich zu diesem Zeitpunkt befindet. Die beiden *Ortsfunktionen* $x(t)$ und $y(t)$ geben bei diesem Modell den Zustand in x- bzw. in y-Richtung an. Die gewöhnlichen Differentialgleichungen

$$\dot{x} = v_x(t) \tag{1.7}$$
$$\dot{y} = v_y(t) \tag{1.8}$$
$$\dot{v}_x = a_x(t) = u_x(t) \tag{1.9}$$
$$\dot{v}_y = a_y(t) = u_y(t) \tag{1.10}$$

beschreiben schließlich die Dynamik des Fahrzeugs. Dies ist nichts anderes als ein System von *Bewegungsgleichungen* aus der klassischen Physik. Es gibt eine Vielzahl komplexerer Dynamiken, diese werden in dieser Arbeit jedoch nicht betrachtet.

Die beiden Gleichungen (1.7) und (1.8) besagen, dass die Ableitung der jeweiligen Ortsfunktion nach der Zeit t der Geschwindigkeit $v(t)$ in die jeweilige Richtung entsprechen muss. Ebenso muss die Beschleunigung $a(t)$ in x- bzw. y-Richtung der Ableitung der jeweiligen Geschwindigkeitsfunktion nach der Zeit t entsprechen. Die Gleichungen (1.9) und (1.10) drücken dies aus.

Die beiden Terme $u_x(t)$ und $u_y(t)$ auf den rechten Seiten dieser Gleichungen bezeichnen die beiden *kontinuierlichen Steuervariablen* des Modells. Sei t_f der, noch zu ermittelnde, Endzeitpunkt der Fahrt. Eine kontinuierliche Stuervariable ist allgemein eine Funktion $u : [0, t_f] \Rightarrow \mathbb{R}^{n_u}$, über sie wird die Bahn des Fahrzeugs gesteuert. Bei dem hier modellierten Fahrzeug wird also die Beschleunigung in x- und y-Richtung kontrolliert.

Komplettiert wird das physikalische Modell durch die *Randbedingungen*

$$x(0) = 0 = x(t_f) \tag{1.11}$$
$$y(0) = 0 = y(t_f) \tag{1.12}$$
$$v_x(0) = 0 = v_x(t_f) \tag{1.13}$$
$$v_y(0) = 0 = v_y(t_f). \tag{1.14}$$

Die beiden Bedingungen (1.11) und (1.12) besagen, dass sich das Fahrzeug sowohl zum Startzeitpunkt t_0 als auch zum Endzeitpunkt t_f im Koordinatenursprung (0,0) befinden muss. Die beiden anderen Bedingungen (1.13) und (1.14) fordern, dass es sich dort im Stillstand, also mit einer Geschwindigkeit von 0, befindet.

Aufbauend auf diesem physikalischen Modell lässt sich nun das Optimierungsproblem formulieren.

Dazu werden zuerst die n zu besuchenden Städte durch ihre (x, y)-Koordinaten $c_1, ..., c_n \in \mathbb{R}^2$ angegeben. Als nächstes wird gefordert, dass jede Stadt zu einem Zeitpunkt $t_i,\ i \in 1, ..., n$ erreicht wird. Als Gleichung lässt sich diese Forderung wie folgt definieren:

$$\begin{pmatrix} x(t_i) \\ y(t_i) \end{pmatrix} - c_i = 0, \qquad i = 1, ..., n. \tag{1.15}$$

Die Reihenfolge, in der die Städte besucht werden, wird analog zum klassischen TSP über Binärvariablen $x_{ij},\ i, j = 1, ..., n$ festgelegt:

$$x_{ij} = \begin{cases} 1 & \text{falls Stadt } i \text{ unmittelbar vor Stadt } j \text{ besucht wird} \\ 0 & \text{sonst} \end{cases}$$

Sie gehen über folgende Nebenbedingungen in die Problemformulierung ein:

$$x_{ij}(t_j - t_i) \geq 0, \qquad \forall\ i, j \in 1, ..., n. \tag{1.16}$$

Es wird also gefordert, dass Stadt j zu einem späteren Zeitpunkt als Stadt i besucht wird, falls $x_{ij} = 1$ gilt. Sind diese Bedingungen sowie die Nebenbedingungen (1.2)-(1.4) des klassischen TSP erfüllt, so ergibt sich eine eindeutige Reihenfolge aller Besuchszeitpunkte t_i. Auf Bedingungen zur Vermeidung von Kurzzyklen kann verzichtet werden, da das physikalische Modell diese bereits kategorisch ausschließt.

Was zur vollständigen Beschreibung des Problems noch fehlt, ist eine passende Zielfunktion, die sowohl die Fahrzeit als auch den Steueraufwand berücksichtigen kann. Dazu müssen als nächstes beide Komponenten definiert werden.

Die Fahrzeit entspricht dabei dem Endzeitpunkt t_f. Etwas schwieriger wird es bei dem gesamten Steueraufwand. Sei zunächst $\mathbf{u}(t) = (u_x, u_y)$ der Vektor aller Steuerungen. Der Steueraufwand zu einem Zeitpunkt t wird nun üblicherweise in Vektorschreibweise als $\mathbf{u}(t)^T\mathbf{u}(t)$ definiert. In Summenschreibweise ist dies

$$\sum_{u_j \in \mathbf{u}(t)} u_j^2 = u_x^2 + u_y^2,$$

also die Summe über die Quadrate der einzelnen Steuerungen. Dieser Term wird auch als *Energieterm* bezeichnet. Um den Steueraufwand für das gesamte Zeitintervall zu bestimmen, wird über $[0, t_f]$ integriert. Zusammen mit zwei Gewichtungsfaktoren a_1 und a_2 ergibt sich folgende Zielfunktion:

$$\text{Minimiere} \qquad F(x_{ij}, u) = a_1 t_f + a_2 \int_0^{t_f} \mathbf{u}(t)^T\mathbf{u}(t)dt. \tag{1.17}$$

Hiermit ist das Problem des motorisierten Handlungsreisenden definiert. Im folgenden Kapitel wird nun der aktuelle Stand der Forschung in Bezug auf Lösungsverfahren für beide Typen von Handlungsreisendenproblemen dargestellt.

Kapitel 2

Stand der Forschung

In diesem Kapitel wird der aktuelle Stand der Forschung bezüglich Lösungsverfahren für klassische als auch motorisierte Handlungsreisendenprobleme dargestellt und daraus die genaue Zielsetzung dieser Arbeit abgeleitet. In Abschnitt 2.1 werden zunächst Lösungsverfahren für klassische TSPe betrachtet. Der Fokus liegt dabei auf den exakten Branch & Bound Verfahren. Die heuristischen Algorithmen werden nur in ihren Grundzügen vorgestellt.

Da motorisierte Handlungsreisendenprobleme zur Klasse der Optimalsteuerungsprobleme gehören, folgt in Abschnitt 2.2 eine Beschreibung der Komponenten zur Lösung dieser. Den Abschluss des Kapitels (2.3) bildet schließlich die Darstellung der exakten Zielsetzung dieser Arbeit.

2.1 Lösung von klassischen TSPe

Klassische Handlungsreisendenprobleme sind *Reihenfolgeprobleme* und gehören somit zur Klasse der kombinatorischen Optimierungsprobleme [Dom95]. Andere Probleme dieser Klasse sind *Zuordnungs-* sowie *Gruppierungsprobleme*. Klassische TSPe fallen zudem in die Klasse der *NP-vollständigen* Probleme. Für diese ist bislang kein Algorithmus bekannt, der eines der Probleme garantiert mit *polynomialem Rechenaufwand* $O(n^x)$ löst.

Man unterscheidet bei den Lösungsverfahren zwischen exakten und heuristischen Verfahren. Exakte Verfahren liefern eine garantiert optimale Lösung, benötigen im Allgemeinen aber viel Rechenzeit und sind nur auf kleine Problemgrößen anwendbar. Auch heuristische Verfahren können eine optimale Lösung liefern, sie können die Optimalität aber nicht feststellen und sie somit auch nicht garantieren [Dom97]. Dafür benötigen sie deutlich weniger Rechenzeit und können folglich auch auf große Problemgrößen angewendet werden.

2.1.1 Heuristische Verfahren

Heuristiken lassen sich nochmals in zwei Gruppen einteilen: *Eröffnungsverfahren* und *Verbesserungs- bzw. lokale Suchverfahren*. Zur heuristischen Lösung eines Problems wird normalerweise eine Kombination von Verfahren aus beiden Gruppen verwendet. Zuerst wird mit einem Eröffnungsverfahren eine möglichst gute zulässige Lösung bestimmt. Lokale Suchverfahren versuchen anschließend, die gefundene Basislösung so zu modifizieren, dass sich der Zielfunktionswert sukzessive verbessert.

Im Laufe der Zeit wurde eine Vielzahl von Eröffnungsverfahren für TSPe entwickelt, eine Übersicht findet sich in [Dom97]. In diesem Abschnitt wird nur die „*sukzessive Einbeziehung*“ beschrieben, da diese implizit in einem der, im nächsten Abschnitt vorgestellten, exakten Verfahren enthalten ist.

Bei diesem Verfahren wird von einem Kurzzyklus ρ mit 2 Knoten ausgegangen, der durch sukzessives Einfügen von weiteren Knoten zu einer zulässigen Rundreise erweitert wird. Jeder neue Knoten wird dabei *bestmöglich* eingefügt, also an der Position wo er die Länge der Tour um den kleinsten Betrag verlängert. Abbildung 2.1 illustriert dies.

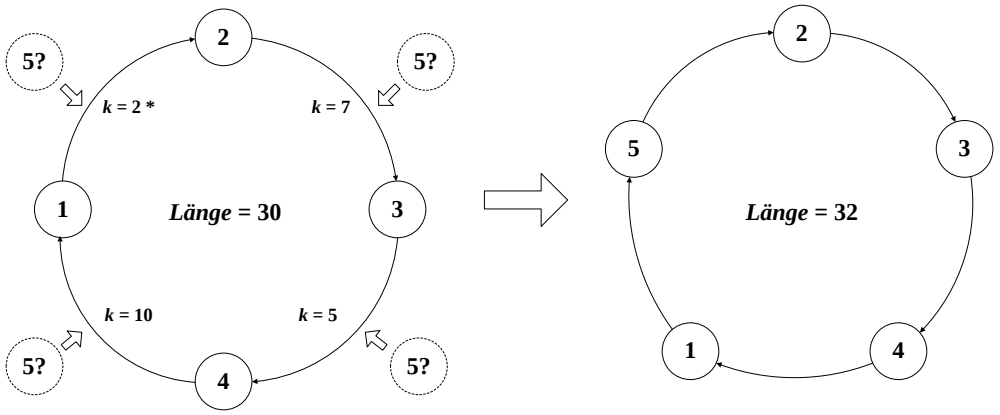


Abbildung 2.1: Sukzessive Einbeziehung

Knoten 5 soll in den Kurzzyklus aus den Knoten 1 bis 4 bestmöglich eingefügt werden, k bezeichnet dabei die Einfügekosten. Knoten 5 wird zwischen Knoten 1 und Knoten 2 eingefügt, da die Kosten hier nur 2 betragen. Die Rundreise verlängert sich genau um die Kosten und hat nach dem Einfügen eine Länge von 32.

Verbesserungsverfahren sind meist Tauschverfahren. Es wird also versucht, die zulässige Rundreise durch den Austausch von Kanten zu verbessern. Der bekannteste Vertreter ist der Lin/Kernighan Algorithmus [LK73], der TSPe in den meisten Fällen nahezu optimal löst. Weitere Ansätze und Verfahren können wiederum [Dom97] entnommen werden.

Besonders verbreitet sind zur Zeit die Meta-Heuristiken *Tabu-Search* und *Simulated Annealing* sowie *genetische Algorithmen*. Der Vorteil dieser Verfahren ist, dass sie mit geringen Anpassungen auf viele verschiedene Problemstellungen angewendet werden können. Codes für die verschiedensten Probleme und in den verschiedensten Programmiersprachen sind frei erhältlich.

2.1.2 Exakte Verfahren

Exakte Lösungsverfahren für TSPe sind entweder Branch & Bound oder Branch & Cut Verfahren. Die Grundidee von beiden Verfahren ist es, ein Ursprungsproblem durch Verzweigung (*Branching*) in mehrere Teilprobleme zu zerlegen und diese zu lösen. Die beiden Verfahren unterscheiden sich darin, wie die Teilprobleme gelöst werden. Im Folgenden wird nun der Ablauf von Branch & Bound Verfahren für Minimierungsprobleme skizziert, eine ausführliche Beschreibung kann wiederum [Dom97] entnommen werden.

Das Verfahren beginnt mit der Bestimmung einer *oberen Schranke* $\overline{F}$ für den optimalen Zielfunktionswert F^*. Dazu wird meist eine Eröffnungsheuristik zur Bestimmung einer zulässigen Lösung des Problems verwendet, der Zielfunktionswert dieser Lösung ist eine erste obere Schranke. Als nächstes wird das ursprüngliche, als P_0 bezeichnete, Problem *relaxiert*. Das relaxierte Problem eines beliebigen Problems P_i wird dabei mit P_i' bezeichnet.

Ein relaxiertes Problem ist ein Ersatzproblem mit einem erweiterten zulässigen Bereich bzw. Lösungsraum. Es gibt viele verschiedene Relaxationsverfahren, das bekannteste ist wohl die *LP-Relaxation*, bei der Ganzzahligkeits-Forderungen in den Nebenbedingungen fallen gelassen werden. Im Rahmen von TSPe werden jedoch meist die Bedingungen zur Vermeidung von Kurzzyklen weggelassen, die Relaxation eines TSP ist folglich ein lineares Zuordnungsproblem.

Als nächstes wird das relaxierte Problem optimal gelöst und überprüft, ob die gefundene Lösung auch eine zulässige Lösung für P_0 ist. Ist sie das, so ist sie auch gleichzeitig eine optimale Lösung und das Verfahren terminiert. Ist sie es nicht, so wird P_0 in zwei oder mehr Teilprobleme P_i verzweigt und diese in eine Kandidatenliste aufgenommen.

Nun werden so lange nach einem bestimmten Schema Probleme aus der Kandidatenliste ausgewählt und anschließend ausgewertet, bis diese leer ist. Eine mögliche Auswahlregel ist die *Minimum Lower Bound* (MLB) Regel. Bei dieser wird immer das Problem mit der kleinsten unteren Schranke gewählt. Jedes Problem wird wiederum relaxiert und optimal gelöst. Der Zielfunktionswert des relaxierten Problems P_i' ist die untere Schranke $\underline{F}_i$ des betrachteten Teilproblems P_i. Die oberen und unteren Schranken werden nun dazu benutzt, den Verzweigungsprozess zu beschränken.

Ein gerade betrachtetes Problem muss dann nicht weiter verzweigt werden, wenn einer der folgenden drei Fälle auftritt:

Fall a: Es gilt $\underline{F}_i \geq \overline{F}$, die bisher gefundene beste Lösung ist also besser als jede in diesem Teilbaum erzielbare Lösung.

Fall b: Die optimale Lösung von P_i' ist eine zulässige Lösung von P_i und es gilt $\underline{F}_i \leq \overline{F}$. In diesem Fall wird $\underline{F}_i := \overline{F}$ gesetzt und die erhaltene Lösung als beste Lösung von P_0 abgespeichert.

Fall c: Es gibt keine zulässige Lösung für P_i'.

Terminiert das Verfahren mit einer leeren Kandidatenliste, so ist die optimale Lösung von P_0 die beste gefundene Lösung mit Zielfunktionswert $\overline{F}$. Die Optimalität wird durch die Traversierung des gesamten Lösungsraumes garantiert.

Ein Branch & Bound Verfahren funktioniert dann am besten, wenn so viele Teilprobleme wie möglich ausgelotet und so wenige wie möglich verzweigt werden müssen. Anhand der drei Fälle des Auslotens ist es leicht ersichtlich, dass gute obere und untere Schranken von großer Bedeutung für die Effizienz von B&B Verfahren sind. Eine Schlussfolgerung ist, dass eine Eröffnungheuristik eine möglichst nahe am Optimum liegende Lösung liefern sollte. Ebenso sollte eine Relaxationstechnik möglichst *scharfe*, also nahe an einer zulässigen Lösung liegende Schranken liefern. Mehr zu diesem Thema folgt in Abschnitt 2.3.

Den Abschluss dieses Abschnitts bildet die Darstellung des B&B Lösungsverfahrens für TSPe von Dakin [Dak97], da dieses Verfahren bei der Implementierung der Verfahren aus Kapitel 3 zum Einsatz kommen wird.

Das Verfahren startet mit $\overline{F} = \infty$, es wird also kein Eröffnungverfahren angewendet um eine erste obere Schranke zu bestimmen. Der grundlegende Ablauf ist ähnlich der sukzessiven Einbeziehung: Jeder Knoten repräsentiert einen Kurzzyklus mit $r \leq n$ Knoten. Auf der r-ten Ebene des Baumes befinden sich dabei alle Kurzzyklen bzw. Subtouren, die r Städte umfassen. Jedes Problem auf Ebene r wird in die $r + 1$ Probleme verzweigt, die durch das Einfügen einer weiteren Stadt an allen möglichen Positionen der Subtour entstehen. Auf der n-ten Ebene finden sich schließlich alle zulässigen Lösungen des TSP.

Angenommen, man befindet sich auf der zweiten Ebene des Baumes und betrachtet gerade den Knoten, der die Subtour {A,B} mit den beiden Städten A und B enthält. Stadt C kann nun an drei möglichen Positionen in die Tour eingefügt werden. Abbildung 2.2 illustriert dies.

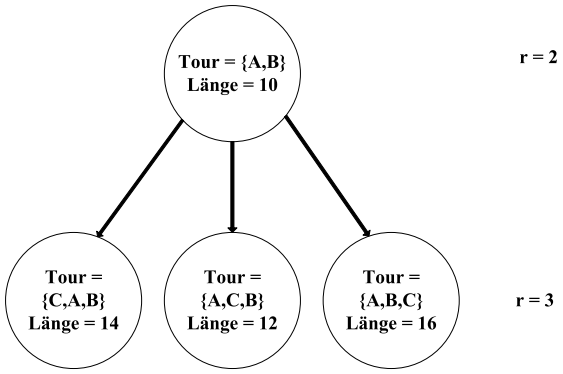


Abbildung 2.2: Verzweigung in Dakins Branch und Bound Verfahren

Das Verfahren verwendet eine Tiefensuche, es wird auf jeder Ebene immer der Knoten mit der kürzesten Tourlänge verzweigt. Im Beispiel aus Abbildung 2.2 wäre dies der Knoten mit der Tour {A,C,B}. Die erste Lösung / obere Schranke ist folglich identisch mit der der sukzessiven Einbeziehung. Zu klären ist nun noch die Frage, wie gut das Verfahren im Vergleich zu anderen B&B Verfahren für asymmetrische TSPe ist.

Betrachtet man die Laufzeit, so dürfte es schlechter als andere Verfahren sein, da Subtouren oft nur auf den unteren Ebenen des Baumes ausgelotet werden können und so relativ viele Knoten untersucht werden können. Zudem finden sich zulässige Lösungen per Definition nur auf der untersten Ebene des Baumes.

Von Vorteil ist dagegen, dass jeder Knoten eine gültige Rundreise für die r enthaltenen Städte darstellt. So können in jedem Teilproblem auch Metriken für die Kosten einer Subtour angewendet werden, bei denen die Kosten der Teilstrecken nicht unabhängig voneinander sind. Dies ist besonders im Hinblick auf den Einsatz von Spline-Kurven in Kapitel 3 von großer Bedeutung. Zudem lässt sich der Algorithmus sehr einfach in Form einer rekursiven Funktion implementieren.

2.2 Lösung von MTSPe

Die in Kapitel 1.2 vorgestellte Formulierung von MTSPe kann nicht direkt gelöst werden, da die Variablen des Problems (Steuerungen) zeitabhängige Funktionen und somit Elemente eines unendlichdimensionalen Raums sind. Zur Optimierung des Problems müssen sie also zunächst durch Elemente eines endlichdimensionalen Raumes approximiert werden. Ein Verfahren, das dies leistet, ist die *direkte Kollokation* [vSG00].

2.2.1 Direkte Kollokation

Ein direktes Kollokationsverfahren transformiert ein Optimalsteuerungsproblem in ein NLP bzw. ein gemischt-ganzzahliges Optimalsteuerungsproblem in ein MINLP. Dazu müssen zwei Typen von Variablen diskretisiert werden: die kontinuierlichen Zustandsvariablen $x(t)$, $y(t)$, $v_x(t)$ und $v_y(t)$ sowie die kontinuierlichen Steuervariablen $u_x(t)$ und $u_y(t)$. Die dazu verwendeten Verfahren werden nun in den Grundzügen vorgestellt, eine ausführliche Darstellung findet sich in [Glo05].

Steuerdiskretisierung

Als es erstes werden mögliche Diskretisierungen der Steuerungen betrachtet. Dazu wird zunächst (das noch nicht bekannte) Zeitintervall $[0, t_f]$ durch $n_{u,\tau}$ Variablen $t_0, t_1, ..., t_{n_{u,\tau}}$ repräsentiert, sie werden auch als *Diskretisierungspunkte* bezeichnet. Für diese Punkte gilt $t_0 < t_1 < ... < t_{n_{u,\tau}}$, ein größerer Index steht also für einen späteren Zeitpunkt.

In den meisten praktischen Arbeiten werden die Steuerungen durch stetige, abschnittsweise lineare Funktionen approximiert. Dazu müssen die Steuerungen an jedem der $n_{u,\tau}$ Diskretisierungspunkte bestimmt werden, folglich benötigt man auch $n_{u,\tau}$ Parameter pro Steuerung. Seien t_i und t_{i+1} zwei beliebige, direkt aufeinanderfolgende Zeitpunkte in $n_{u,\tau}$ und seien $u(t_i)$, $u(t_{i+1})$ die korrespondierenden Steuerungen. Die Steuerung für einen beliebigen Zeitpunkt t im Intervall $[t_i, t_{i+1}]$ erhält man bei dieser Variante durch einfache lineare Interpolation zwischen $u(t_i)$ und $u(t_{i+1})$.

Zustandsdiskretisierung

Nachdem die kontinuierlichen Steuerungen diskretisiert sind, müssen nun auch noch die Zustände $x(t)$, $y(t)$, $v_x(t)$ und $v_y(t)$ diskretisiert werden. Sei dazu o.B.d.A. $x(t)$ der zu diskretisierende Zustand. Auch hier wir das Zeitintervall $[0, t_f]$ durch $n_{x,\tau}$ Diskretisierungspunkte approximiert. Es wird dabei nicht zwangsläufig $n_{x,\tau} = n_{u,\tau}$ gefordert, die Auflösung der Steuerungs- und der Zustandsdiskretisierung kann sich also unterscheiden.

Welche Bedingungen muss die den Zustand $x(t)$ approximierende Funktion $x_{app}(t)$ erfüllen? Da die Differentialgleichungs-Nebenbedingungen eine zweimalige Differenzierbarkeit von x nach t fordern, muss $\dot{x}(t) = v_x(t)$ stetig sein. Somit muss $x_{app}(t)$ also *stetig differenzierbar* sein.

Approximationsfunktionen mit diesen Eigenschaften sind die, in den meisten praktischen Arbeiten verwendeten, *kubischen Splines*. Sie benötigen für eine stetig differenzierbare Approximation lediglich $2n_{x,\tau}$ Parameter. Die Parameter sind die Werte sowie Steigungen an jedem Diskretisierungspunkt $t_i,\ i = 1, ..., n_{x,\tau}$. Abbildung 2.3 illustriert die Diskretisierung der Zeit sowohl für die Steuerungen als auch die Zustände. In der Darstellung gilt hier $n_{x,\tau} = n_{u,\tau}$, die Zeit ist also für beide Typen gleich fein aufgelöst.

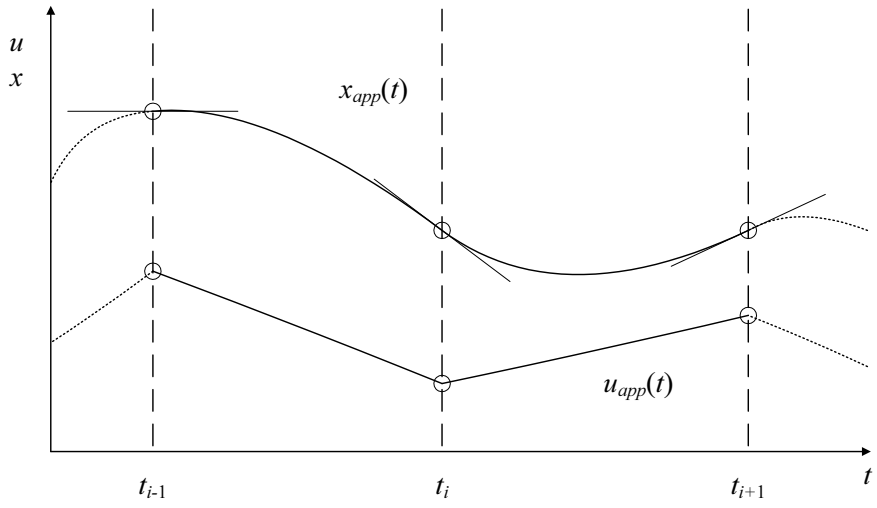


Abbildung 2.3: Zeitdiskretisierung und Approximationsfunktionen

Kollokationsbedingungen

Da nun sowohl die Steuerungen, als auch die Zustände diskretisiert sind, können nun die *Kollokationsbedingungen* für das MTSP definiert werden. Diese stellen sicher, dass die Differentialgleichungen des physikalischen Modells an ausgewählten Zeitpunkten, den sogenannten *Kollokationspunkten* τ_j, erfüllt sind. Aus jeder der Differentialgleichungs-Nebenbedingungen (1.7)-(1.10) des Optimalsteuerungsproblems wird so ein System von Gleichungsnebenbedingungen:

$$\dot{x}(\tau_j) = v_x(\tau_j), \quad \forall \tau_j \tag{2.1}$$
$$\dot{y}(\tau_j) = v_y(\tau_j), \quad \forall \tau_j \tag{2.2}$$
$$\dot{v}_x(\tau_j) = u_x(\tau_j), \quad \forall \tau_j \tag{2.3}$$
$$\dot{v}_y(\tau_j) = u_y(\tau_j), \quad \forall \tau_j \tag{2.4}$$

Damit das Gleichungssystem eine eindeutige Lösung besitzt, werden pro Intervall $[t_i, t_{i+1}]$ bei einer stetig differenzierbaren, stückweise polynomialen Funktion $x_{app}(t)$ mit Polynomstücken vom Grad k mindestens $k-1$ Kollokationspunkte benötigt. Die relative Lage der Kollokationspunkte zueinander ist in jedem Intervall gleich. Doch wie genau sollten die Punkte in dem Intervall liegen? Man unterscheidet hier zwischen *Gauß-Punkten* und *Lobatto-Punkten*, eine Erklärung der beiden Methoden findet sich in [Glo05].

Damit ist das direkte Kollokationsverfahren beendet. Das Ergebnis seiner Anwendung auf die ursprüngliche Formulierung des MTSP aus Kapitel 1.2 ist ein gemischtganzzahliges, nicht lineares Optimierungsproblem. Im nächsten Abschnitt werden nun Lösungsverfahren für diesen Typ von Optimierungsproblemen dargestellt.

2.2.2 Branch & Bound Verfahren für MTSPe

Bevor in diesem Abschnitt nun auf die Lösungsverfahren für gemischt-ganzzahliges nichtlineare Optimierungsprobleme eingegangen wird, werden sie zunächst noch einmal formal definiert. Ein allgemeines MINLP hat nach [Ley93] die Form

$$\text{Minimiere } f(x, y) \tag{2.5}$$

unter den Nebenbedingungen

$$g(x, y) \leq 0 \tag{2.6}$$
$$x \in X \tag{2.7}$$
$$y \in Y \text{ und ganzzahlig}, \tag{2.8}$$

wobei f die nichtlineare Zielfunktion ist und g die nichtlinearen Nebenbedingungen sind. Die Mengen X und Y sind durch einfache obere und untere Schranken der Form $a \leq x \leq b$ bestimmt.

Um ein MINLP zu lösen, müssen Lösungsverfahren für ganzzahlige MIPs und nichtlineare NLPs kombiniert werden. Dazu wird in dieser Arbeit kurz das in [Glo05] verwendete Branch & Bound Verfahren vorgestellt. Eine Übersicht über andere Lösungsverfahren wie *Outer Approximation* oder *erweiterte Schnittebenenverfahren* sowie weiterführende Literatur findet sich in [BP03].

Der Kern des hier vorgestellten Verfahrens ist eine *NLP-Relaxation* des MINLPs in jedem Knoten des Baums. Die Forderung nach der Ganzzahligkeit der Binärvariablen x_{ij} aus Kapitel 1.2 wird dabei fallen gelassen. Formal wird die Nebenbedingung $x_{ij} \in \{0, 1\}$ also zu $0 \leq x_{ij} \leq 1$, das resultierende Problem ist ein reines NLP und kann mit den entsprechenden Lösungsverfahren gelöst werden.

Zusammenfassend kann das Verfahren wie folgt beschrieben werden [BGH+02]:

1. Bestimme mit einer Heuristik zulässige x_{ij}. Löse anschließend das sich ergebende NLP, um eine globale obere Schranke $\overline{F}$ zu erhalten.
2. Führe in der Wurzel eine NLP-Relaxation durch und löse das sich ergebende NLP. Man erhält eine erste untere Schranke $\underline{F}_0$.
3. Bestimme die zu verzweigende Binärvariable $x_{i,j}$ und fixiere sie auf 0 bzw. 1. Löse beide relaxierten Subprobleme um zwei neue Knoten zu erhalten.
4. Bestimme den nächsten zu verzweigenden Knoten. Hier kann entweder eine Tiefen-, eine Breitensuche oder die *Minimum Lower Bound* (MLB) Regel zum Einsatz kommen.
5. Lote einen Knoten aus, wenn die untere Schranke größer als die bisher beste obere Schranke ist oder wenn die NLP-Relaxation keine Lösung hat.
6. Ersetzen der aktuell besten Lösung $\overline{F}$ und der oberen Schranke wenn eine zulässige Lösung mit $\underline{F}_i \leq \overline{F}$ gefunden wurde.

Es handelt sich hierbei faktisch um das Branch & Bound Verfahren von Little et al. [LMSK63] für klassische, asymmetrische TSPe. Andere B&B Algorithmen für klassische TSPe, wie z.B. Varianten von *Subtoureliminationsverfahren* [Dom97], können nicht angewendet werden, da Subtouren (Kurzzyklen) bei MTSPe prinzipiell nicht vorkommen können. Als nächstes werden nun Lösungsverfahren für NLPs diskutiert.

2.2.3 Lösung der NLP-Relaxationen

Nichtlineare Optimierungsprobleme können numerisch mit *sequentieller quadratischer Programmierung* (SQP) gelöst werden [GvS02], einem gegenüber der mathematischen Problemstruktur flexiblem Optimierungsverfahren. Um es erläutern zu können, ist zunächst die Darstellung der Optimalitätsbedingungen erforderlich.

Optimalitätsbedingungen

Zur Herleitung der Bedingungen wird zunächst die Lagrangefunktion definiert. Sei $\boldsymbol{x}$ der Vektor aller Variablen und Φ die Zielfunktion des NLP. Dazu seien $\boldsymbol{g}$ die Gleichungs- und $\boldsymbol{h}$ die Ungleichungs-Nebenbedingungen, n_g bzw. n_h bezeichnen die Anzahl der Bedingungen. Der zulässige Bereich des Problems wird durch $\mathbb{Z}$ repräsentiert.

Definition 2.1 (Lagrangefunktion)**.**
Die Linearkombination aus Zielfunktion und Nebenbedingungen

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \Phi(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{\mu}^T \boldsymbol{h}(\boldsymbol{x})$$

heißt Lagrangefunktion des NLP. Die reellen Parameter $\boldsymbol{\lambda} \in \mathbb{R}^{n_g}$ und $\boldsymbol{\mu} \in \mathbb{R}^{+n_h}$ werden *Lagrange'sche Multiplikatoren* genannt.

Auf Basis einer Lagrangefunktion können nun die *Karush-Kuhn-Tucker* Optimalitätsbedingungen für eine Lösung $\boldsymbol{x}^*$ definiert werden:

$$\boldsymbol{x}^* \in \mathbb{Z} \tag{2.9}$$

$$L_{\boldsymbol{x}}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = 0, \qquad \boldsymbol{\mu}^* \geq 0 \tag{2.10}$$

$$\mu_j^* \neq 0 \Leftrightarrow h_j(\boldsymbol{x}_*) = 0, \qquad j = 1, ..., n_h \tag{2.11}$$

Die erste Gleichung (2.9) fordert zunächst die *primale Zulässigkeit*, $\boldsymbol{x}^*$ muss also alle Nebenbedingungen erfüllen. Sei nun $\mathbb{A}(\boldsymbol{x})$ eine Indexmenge, die die Indizes aller aktiven Ungleichungs-Nebenbedingungen h_j enthält. Eine Nebenbedingung wird dann als *aktiv* bezeichnet, wenn sie mit Gleichheit erfüllt ist. Mit Hilfe dieser Definition kann nun die Bedeutung der beiden nächsten Bedingungen anschaulich erläutert werden.

Die *Multiplikatorregel* (2.10) bedeutet zusammen mit der *strikten Komplementarität* (2.11), dass der negative Gradient der Zielfunktion $-\nabla\Phi$ in dem von den Gradienten der *aktiven Nebenbedingungen* $\nabla g_i, i = 1, ..., n_g$ und $\nabla h_j, j \in \mathbb{A}(\boldsymbol{x})$ aufgespannten Kegel liegt. Ein Fortschreiben in Richtung sinkender Zielfunktionswerte würde folglich zum Verlassen des zulässigen Bereichs $\mathbb{Z}$ führen. Abbildung 2.4 illustriert dies exemplarisch für zwei aktive Ungleichungs-Nebenbedingungen $g(\boldsymbol{x})$ und $h(\boldsymbol{x})$, der zulässige Bereich ist schraffiert dargestellt.

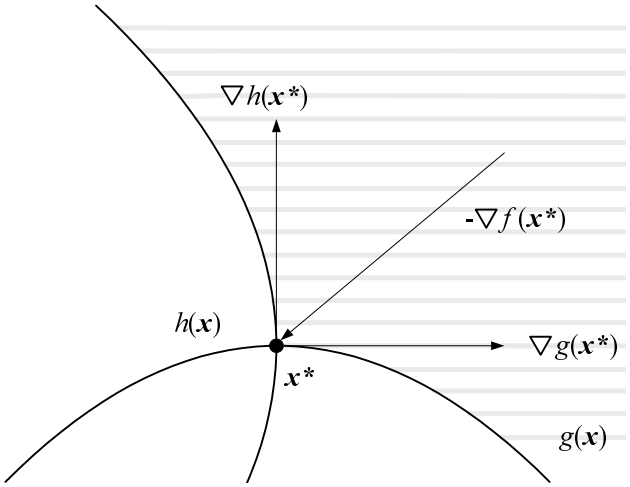


Abbildung 2.4: Karush-Kuhn-Tucker Bedingungen

Als Bedingungen erster Ordnung sind die Karush-Kuhn-Tucker Bedingungen *notwendige* Bedingungen für ein *lokales Optimum*. Hinreichende Optimalitätsbedingungen für die lokale Optimalität einer Lösung $\boldsymbol{x}_*$ ergeben sich in Kombination mit den folgenden Bedingungen zweiter Ordnung [Glo05]:

$$s^T L_{\boldsymbol{xx}}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) s > 0, \ \forall \boldsymbol{s} \in \mathbb{S} \subset \mathbb{R}^{n_y} \tag{2.12}$$

$$\sum_{i=0}^{n_g} \alpha_i \nabla g_i(\boldsymbol{x}^*) + \sum_{i \in \mathbb{A}} (\boldsymbol{x}^*) \beta_i \nabla h_j(\boldsymbol{x}^*) \neq 0, \ \forall \alpha_i, \beta_i \neq 0 \tag{2.13}$$

Gleichung (2.12) fordert, dass die Hesse-Matrix (enthält alle partiellen Ableitungen zweiter Ordnung) der Lagrangefunktion L im Lösungspunkt $\boldsymbol{x}^*$ tangential zu den Gleichungs-Nebenbedingungen g_i und positiv definit zu den aktiven Ungleichungs-Nebenbedingungen h_j ist. In einfachen Worten bedeutet dies, dass die Zielfunktionswerte steigen, sobald man sich im zulässigen Bereich von $\boldsymbol{x}^*$ weg bewegt.

Die letzte Gleichung (2.13) stellt schließlich eine Regularitätsbedingung dar. Sie fordert, dass die Gradienten aller aktiven Nebenbedingungen im Lösungspunkt $\boldsymbol{x}^*$ linear unabhängig sind.

Dies sind die vollständigen Optimalitätsbedingungen für ein *lokales Optimum* eines NLP, wie bestimmt man nun aber ein *globales Optimum*? Am einfachsten lässt sich diese Frage für ein *konvexes* NLP beantworten. Bei diesem Typ von Problemen ist der zulässige Bereich $\mathbb{Z}$ konvex, folglich gibt es nur ein einziges lokales Optimum, das gleichzeitig auch das globale Optimum darstellt.

Sequentielle Quadratische Programmierung (SQP)

Auf Basis dieser Optimalitätsbedingungen kann nun das Verfahren der sequentiellen quadratischen Programmierung beschrieben werden. Prinzipiell handelt es sich dabei um ein iteratives Verfahren, ähnlich dem Newton-Raphson Algorithmus. Ausgehend von einer Iterierten $\boldsymbol{x}_k$ wird die nächste Iterierte $\boldsymbol{x}_{k+1}$ bestimmt, bis Konvergenz erreicht wird [GMS02].

Das Verfahren beginnt mit einer *Startschätzung* $\boldsymbol{x}_0$. Zu dieser Startschätzung wird als nächstes ein *quadratisches Problem* (QP) mit linearen Nebenbedingungen formuliert, das die Lagrange Funktion L des NLP approximiert und die gleichen Optimalitätsbedingungen zweiter Ordnung hat.

Als nächstes wird das QP mit den Standardverfahren für diese Problemklasse gelöst. Die Lösung wird mit $\boldsymbol{x}_0^*$ bezeichnet, sie liefert entweder direkt die nächste Iterierte $\boldsymbol{x}_1$ oder zumindest eine gute Suchrichtung für diese.

Steht die Suchrichtung, so muss noch die Schrittweite bestimmt werden, also wie weit man sich vom aktuellen Iterationspunkt in die Suchrichtung bewegt. Dazu kommen meist *Strafkosten-* oder *Penaltyfunktionen* zum Einsatz, die auf der Lagrangefunktion basieren. Das Ergebnis ist die nächste Iterierte $\boldsymbol{x}_{k+1}$. Das Verfahren stoppt, sobald eine Iterierte die Optimalitätsbedingungen (2.9)-(2.13) erfüllt.

Um das Verfahren zu illustrieren, wird ein NLP mit nur zwei Problemvariablen x_1 und x_2 betrachtet, die zusammen den Vektor $\boldsymbol{x} = (x_1, x_2)$ bilden. Außer der nichtlinearen Zielfunktion $f(x)$ gibt es noch eine nichtlineare Ungleichungs-Nebenbedingung $g_1(x)$ sowie eine lineare Ungleichungs-Nebenbedingung $g_2(x)$. Abbildung 2.5 zeigt die Suche nach einer optimalen Lösung ausgehend von der Startschätzung $x_0 = (2, 0)$. Nach sechs äußeren Iterationen terminiert das Verfahren mit der optimalen Lösung $x^* = (0, -3)$. Der zulässige Bereich liegt innerhalb des von $g_1(x) = 0$ definierten Kreises sowie unterhalb der durch $g_2(x) = 0$ definierten Gerade.

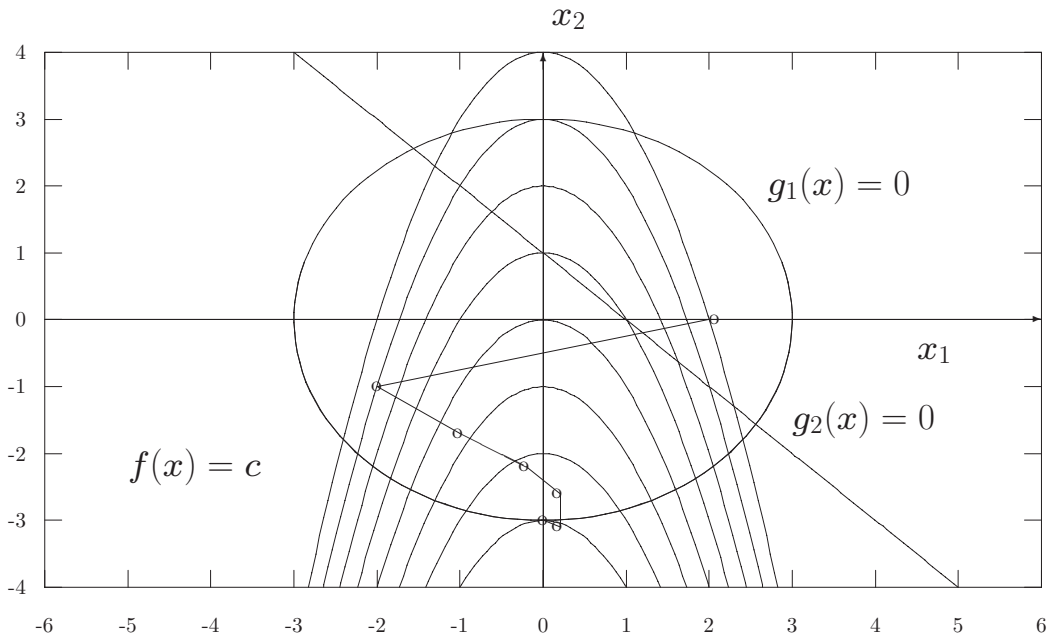


Abbildung 2.5: Lösung eines NLP mit SQP [Sch04]

Es existieren viele verschiedene Implementationen von SQP Verfahren, eine Übersicht findet sich in [Fou05]. Wie gut diese funktionieren ist von der Struktur des zu lösenden NLP abhängig. Um einen passenden *Löser* (engl. *Solver*) auszuwählen, muss also zunächst die Struktur der von direkten Kollokationsverfahren erzeugten NLPs beschrieben werden.

Diese haben nach [GvS02] die folgenden Eigenschaften:

- Es handelt sich um große NLPs mit vielen Variablen und vielen Nebenbedingungen
- Viele Nebenbedingungen sind am Lösungspunkt aktiv, z.B. alle aus der Kollokation resultierenden Gleichungsnebenbedingungen. Die Anzahl der freien Variablen ist folglich deutlich kleiner als die Anzahl aller Variablen.
- Die Jacobi-Matrizen des NLP sind dünn besetzt und strukturiert. Nur wenige Prozent der Einträge sind ungleich 0, zudem verringert sich der Prozentsatz, je feiner das Diskretisierungsgitter gewählt wird.

Die meisten SQP Solver gehen dagegen von vollbesetzten Jacobi-Matrizen aus. Sie können die durch direkte Kollokation erzeugten NLPs zwar lösen, benötigen dafür aber sehr viel Zeit. Der SNOPT Solver [GMS02] ist dagegen auf Probleme mit dünn besetzten und strukturierten Jacobi-Matrizen optimiert und ist bei so strukturierten Problemen teilweise um den Faktor 100 schneller als andere SQP Solver.

2.3 Auswirkungen guter Schranken und Startschätzungen auf die Laufzeit der Verfahren

Branch & Bound Verfahren starten meist mit einer oberen Schranke $\overline{F}$, die aus einer zulässigen Lösung des ursprünglichen Problems bestimmt wird. SQP-Verfahren benötigen eine Startschätzung der optimalen Lösung, um überhaupt arbeiten zu können. Welche Auswirkungen hat also die Güte der zulässigen Lösung / der Startschätzung auf das Laufzeitverhalten der Algorithmen?

Bei Branch & Bound Verfahren haben sowohl die obere Schranke $\overline{F}$, als auch die unteren Schranken $\underline{F}_i$ fundamentale Auswirkungen auf das Laufzeitverhalten, da anhand von ihnen entschieden wird, ob Knoten in der Kandidatenliste ausgelotet werden können oder nicht. Sind beide Schranken gut bzw. *scharf*, so können viele Knoten bereits in den oberen Ebenen des Baumes ausgelotet werden und müssen folglich nicht verzweigt und ausgewertet werden.

Auch bei SQP Verfahren hat die Startschätzung $\boldsymbol{x}_0$ Auswirkungen auf das Laufzeitverhalten. Liegt sie bereits in der Nähe des Optimums, so werden in der Regel nur wenige Iterationen bis zur Konvergenz benötigt. Ist sie schlecht, so werden im besten Fall viele Iterationen benötigt, im schlimmsten Falle konvergiert das Verfahren überhaupt nicht.

Im Falle eines nicht-konvexen NLP ist eine gute Startschätzung von noch größerer Bedeutung, da das SQP-Verfahren zu *lokalen Optima* konvergiert. Liegt das geschätzte $\boldsymbol{x}_0$ also nicht in der näheren *Umgebung* des globalen Optimums, so ist es sehr wahrscheinlich, dass SQP in einem anderen lokalen Optimum stoppt und die optimale Lösung nicht gefunden wird.

Gute Schranken bzw. Startschätzungen senken also die Laufzeit der beiden Verfahren. Die eigentliche Zielsetzung ist es jedoch, die *Gesamtlaufzeit* zu minimieren, also der Summe aus der Laufzeit des Verfahrens und der Laufzeit zur Generierung der Schranken bzw. Schätzungen. Gesucht sind folglich Verfahren zur Bestimmung von Schranken und Startschätzungen, die möglichst wenig Rechenzeit benötigen und dabei möglichst scharfe Ergebnisse liefern.

Das Ziel dieser Arbeit ist es einerseits, Verfahren zur Generierung von Startschätzungen für den kombinatorischen Anteil von motorisierten Handlungsreisendenproblemen zu entwickeln und sie auf Laufzeitverhalten und Lösungsgüte zu untersuchen. Andererseits sollen auch Approximationsverfahren für die nichtlinearen Teilprobleme entwickelt und beurteilt werden. Als Basis für die Entwicklung dienen Daten aus dem vollwertigen MTSP.

Im folgenden Kapitel 3 werden die entwickelten Verfahren zunächst theoretisch fundiert dargestellt und anschließend die experimentellen Ergebnisse beurteilt.

Kapitel 3

Ansätze zur Bestimmung von Basislösungen und Schranken

Ziel dieses Kapitels ist es, Verfahren zur Bestimmung von unteren Schranken für die nichtlinearen Teilprobleme eines MTSP sowie zur Bestimmung von Basislösungen für den kombinatorischen Anteil vorzustellen sowie zu analysieren. Die Algorithmen werden dabei theoretisch fundiert beschrieben und, soweit möglich, graphisch illustriert. Auf die Darstellung von Programmcode wird verzichtet.

Splines bilden die Grundlage für fast alle Verfahren, sie werden in Kapitel 3.1 zunächst vorgestellt. Nach einer kurzen und allgemein gehaltenen Einführung in diese Klasse von parametrischen Kurven wird auf die in dieser Arbeit verwendeten Typen und deren relevanten Eigenschaften eingegangen.

In Abschnitt 3.2 wird zunächst gezeigt, wie sich Splines zur Bestimmung einer Basislösung des kombinatorischen Anteils von MTSPs einsetzen lassen. Dazu kommt das Branch & Bound Verfahren von Dakin zum Einsatz, als Kostenfunktion wird ein Index auf Basis der durchschnittlichen Krümmung der Kurve sowie der Bogenlänge verwendet.

Der folgende Abschnitt 3.3 wird sich mit der Integration eines physikalischen Fahrzeugmodells in eine bestehende Spline-Kurve auseinander setzen. Von besonderer Bedeutung ist dabei die Bestimmung der Zeit, die ein gegebenes Fahrzeug zum Abfahren eines gegebenen Splines benötigt. Dazu muss die Parametrisierung eines Splines in eine reale Fahrzeit transformiert werden. Aus diesem Ansatz wird eine Approximation der Fahrzeit für Probleme ohne Geschwindigkeitsbeschränkung abgeleitet.

Den Abschluss des Kapitels bildet die Herleitung und Analyse beweisbarer unterer Schranken für die Fahrzeit. In Abschnitt 3.4 werden Probleme mit, in Abschnitt 3.5 Probleme ohne Geschwindigkeitslimit betrachtet.

3.1 Spline-Kurven

Spline-Kurven gehören zur Klasse der *parametrischen Kurven*. Allgemein beschreibt eine parametrische Kurve Punkte in einem n-dimensionalen Raum als Funktion eines Parameters t. Mathematisch gesprochen handelt es sich um eine stetige Funktion $\boldsymbol{p} : t \rightarrow \boldsymbol{p}(t), t \in [a, b],\ p(t) \in \mathbb{R}^n$, die für jeden gültigen Wert von t einen Punkt im n-dimensionalen Raum $\mathbb{R}^n$ liefert [AMH02] [DB78].

Ein einfaches Beispiel ist die Funktion $\boldsymbol{p}(t) = t\boldsymbol{p}_0 + (1-t)\boldsymbol{p}_1, t \in [0, 1]$. Sind $\boldsymbol{p}_0$ und $\boldsymbol{p}_1$ zwei Punkte im Raum, so beschreibt $\boldsymbol{p}(t)$ die lineare Interpolation zwischen diesen beiden Punkten. Sei nun eine ganze Folge von n Punkten $\boldsymbol{p}_0, ..., \boldsymbol{p}_n$ gegeben, gesucht sei dazu eine interpolierende Kurve, die die Punkte in der gegebenen Reihenfolge verbindet.

Eine einfache und nahe liegende Lösung ist es, auf jedes Paar von Punkten $(\boldsymbol{p}_i, \boldsymbol{p}_{i+1})$ die eben vorgestellte lineare Interpolationsfunktion anzuwenden. Das Ergebnis ist eine stetige, aber nicht stetig differenzierbare Kurve, die sich *abschnittsweise* aus linearen Funktionen bzw. Polynomen mit Grad 1 zusammensetzt.

Eine solche interpolierende Kurve, die sich stückweise aus Polynomen mit maximalem Grad k zusammensetzt, wird als *Spline k-ten Grades* bezeichnet. Der Begriff *Spline* stammt aus dem Englischen und bezeichnet eine dünne, im Schiffbau verwendete Holzlatte. Wird diese mit Nägeln am Schiffsrumpf befestigt, so krümmt sie sich exakt so wie ein kubischer Spline (maximaler Polynomgrad $k = 3$), den man durch die Fixpunkte legt.

Für diese Arbeit werden Splines benötigt, die nicht nur stetig, sondern auch stetig differenzierbar sind. Nur solche Kurven lassen in Kombination mit dem physikalischen Bewegungsmodell auf gute Näherungen hoffen. Aus diesem Grund werden in folgenden Abschnitten verschiedene Spline-Typen vorgestellt, die diese Anforderungen erfüllen.

3.1.1 Kubische Splines

Ein zwischen $n + 1$ Punkten interpolierender kubischer Spline setzt sich aus n Segmenten zusammen, die jeweils durch ein kubisches Polynom $\boldsymbol{p}(t) = x_0 + x_1 t + x_2 t^2 + x_3 t^3, t \in [0, 1]$ beschrieben werden. Um eine Spline-Kurve zu erhalten sind also insgesamt $4n$ Parameter zu bestimmen.

Bevor auf die Bestimmung der Parameter eingegangen wird, wird zunächst kurz ein Stetigkeitsmaß für Splines vorgestellt. Sei dazu $S_i(t), t \in [0, 1]$ das Kurvensegment zwischen $\boldsymbol{p}_i$ und $\boldsymbol{p}_{i+1}$. Eine Kurve wird als C^m stetig bezeichnet, wenn sie in jedem Punkt m-mal differenzierbar ist.

Für sich betrachtet ist jedes Segment C^∞ stetig, der maximale Polynomgrad k bestimmt jedoch die Stetigkeit an den Verbindungspunkten zwischen Segmenten und somit die Stetigkeit der gesamten Kurve. Ein linearer Spline ($k = 1$) ist beispielsweise als ganzes betrachtet nur C^0 stetig. Dies liegt daran, dass die ersten Ableitungen

der an einem Punkt $\boldsymbol{p}_i$ aufeinander treffenden Segmente in diesem Punkt nicht übereinstimmen, also $S_i'(1) \neq S_{i+1}'(0)$ gilt.

Diese Aussage gilt analog für alle $k \geq 1$. Für einen insgesamt C^m stetigen Spline werden Polynomsegmente benötigt, die *mindestens* vom Grad $m+1$ sind. Für eine stetig differenzierbare Kurve (C^1 stetig) benötigt man folglich mindestens Polynome vom Grad 2 bzw. *quadratische Polynome*.

Die Bestimmung der Parameter eines eindimensionalen kubischen Splines mit $n+1$ Punkten geschieht auf Basis der folgenden Bedingungen [Wei02]:

- Der Spline muss durch jeden Punkt $\boldsymbol{p}_i, i = 1, ..., n+1$ gehen ($2n$ Bedingungen)
- Die ersten Ableitungen müssen an jedem inneren Punkt $\boldsymbol{p}_j, j = 2, ..., n$ übereinstimmen ($n-1$ Bedingungen)
- Die zweiten Ableitungen müssen an jedem inneren Punkt $\boldsymbol{p}_j, j = 2, ..., n$ übereinstimmen ($n-1$ Bedingungen)

Man erhält also insgesamt $2n + (n-1) + (n-1) = 4n - 2$ Bedingungen für $4n$ Parameter. Die beiden noch fehlenden Bedingungen, auch *Abschlussbedingungen* genannt, lassen sich nun auf viele verschiedene Arten ergänzen, in dieser Arbeit wird nur die benötigte Variante sowie zum Vergleich eine weitere vorgestellt.

Das physikalische Modell aus Kapitel 1.2 enthält mit den Gleichungen (1.13) und (1.14) zwei Randbedingungen, die fordern, dass die Geschwindigkeit des Fahrzeugs im ersten und letzten Punkt 0 ist. Interpretiert man nun einen zweidimensionalen Spline $s(t) : t \to s(t),\ s(t) \in \mathbb{R}^2$ als Ortsfunktion eines physikalischen Modells, so muss auch dieser die Randbedingungen erfüllen.

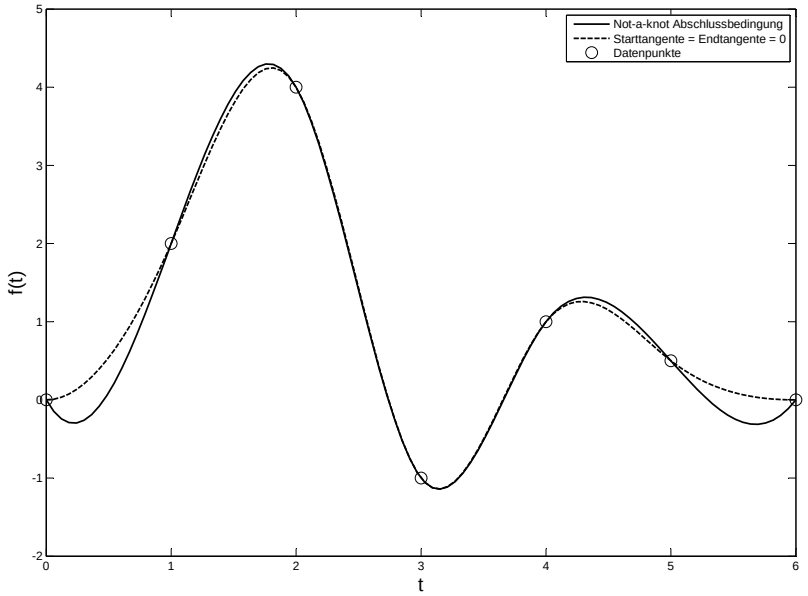


Abbildung 3.1: Abschlussbedingungen

Diese Randbedingungen liefern die beiden fehlenden Bedingungen für den kubischen Spline: Man fordert einfach, dass die erste Ableitung der Kurve im ersten Punkt $\boldsymbol{p}_1$ und im letzten Punkt $\boldsymbol{p}_{n+1}$ den Wert 0 hat. Mit diesen Bedingungen ergibt sich ein tridiagonales lineares Gleichungssystem, das effizient gelöst werden kann.

Eine zweite Variante ist die so genannte *not-a-knot* Bedingung. Sie fordert, dass im ersten ($\boldsymbol{p}_2$) und letzten ($\boldsymbol{p}_n$) inneren Punkt auch die dritten Ableitungen übereinstimmen. In anderen Worten läuft der Spline in einer geraden Linie durch diese Punkte. In Abbildung 3.1 sind beide Bedingungen gegenübergestellt, ein Überblick über weitere Abschlussbedinungen findet sich in [Mat02].

3.1.2 Kubische Hermite Splines

Kubische Splines haben den Nachteil, dass sich ihre Form nur sehr eingeschränkt durch die Abschlussbedingungen beeinflussen lässt. Dazu kommt noch, dass die Veränderung eines Parameters, sei es ein Punkt $\boldsymbol{p}_i$ oder eine vorgegebene Start- bzw. Endtangente, den Verlauf der gesamten Kurve verändert.

Die durchgezogene Linie in Abbildung 3.2 zeigt einen kubischen Spline, der durch durch die Punkte $(0, 2, 4, -1, 1, 0.5, 0)$ geht. Verändert man nun die Position des vierten Punktes von -1 auf -2 und legt erneut einen kubischen Spline durch die Punkte, so ergibt sich die gestrichelte Linie, deren Verlauf sich in jedem Segment von dem der ersten Kurve unterscheidet. Lokale Veränderungen wirken sich folglich global aus, sagt man daher auch, dass die Kurve einen *globalen Träger* hat.

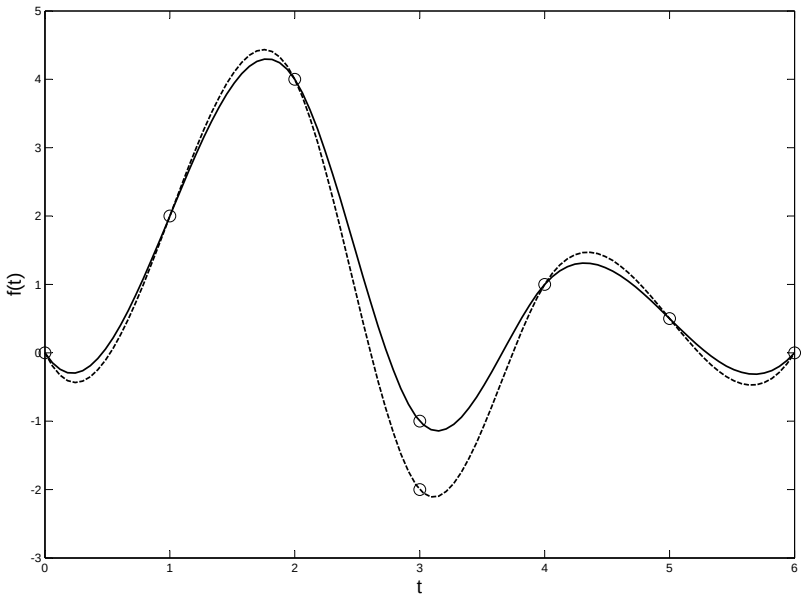


Abbildung 3.2: Die Eigenschaft eines globalen Trägers

Kardinale Splines sind nun Splines mit einem *lokalen Träger*, eine lokale Veränderung wirkt sich also nur in begrenztem Umfang auf den Verlauf der gesamten Kurve aus.

Eine geeignete Basis dafür ist die *Hermite-Basis* [Hei98]. Sie ist auf dem Intervall $t \in [0,1]$ wie folgt definiert:

$$\begin{aligned} h_0(t) &= 2t^3 - 3t^2 + 1 \\ h_1(t) &= t^3 - 2t^2 + t \\ h_2(t) &= t^3 - t^2 \\ h_3(t) &= -2t^3 + 3t^2 \end{aligned}$$

Der Verlauf jedes Segments $S_i(t)$ eines kubischen Hermite Spline wird durch vier Parameter bestimmt: Die Anfangs- und Endpunkte $\boldsymbol{p}_0$ und $\boldsymbol{p}_1$ sowie den Tangenten im Anfangs- und Endpunkt $\boldsymbol{m}_0$ und $\boldsymbol{m}_1$. Die Interpolationsfunktion ist

$$\boldsymbol{p}(t) = \boldsymbol{p}_0 h_0 + \boldsymbol{m}_0 h_1 + \boldsymbol{m}_1 h_2 + \boldsymbol{p}_1 h_3, \quad t \in [0,1]. \tag{3.1}$$

Die Tangenten bieten dem Anwender direkte und lokale Kontrolle über den Verlauf der Kurve, der Umgang mit ihnen ist jedoch wenig intuitiv. Aus diesem Grund wurden mehrere Verfahren zur Berechnung der Tangenten auf Basis weniger und intuitiver Parameter entwickelt. In den folgenden Unterabschnitten werden nun die beiden wichtigsten Berechnungsverfahren kurz vorgestellt.

Kardinale Splines

Als *kardinale Splines* werden kubische Hermite Splines bezeichnet, bei denen die Tangente für einen Punkt $\boldsymbol{p}_i$ auf Basis des vorherigen Punkts $\boldsymbol{p}_{i-1}$, des nachfolgenden Punkts $\boldsymbol{p}_{i+1}$, sowie einem Spannungsparameter $s \in [-1,1]$ berechnet wird. Daraus folgt, dass kardinale Splines maximal C^1 stetig sind, denn für C^2 Stetigkeit müssten auch die Punkte $\boldsymbol{p}_{i-2}$ und $\boldsymbol{p}_{i+2}$ herangezogen werden. Die genaue Formel zur Berechnung einer zu $\boldsymbol{p}_i$ gehörenden Tangente $\boldsymbol{m}_i$ ist

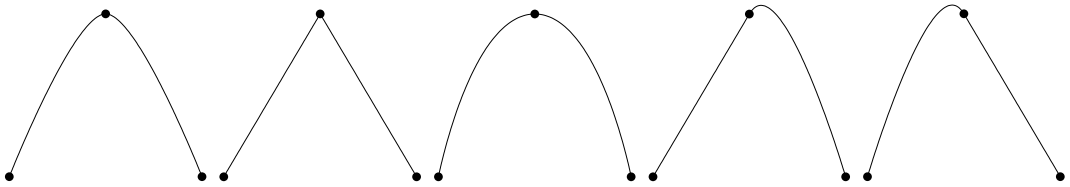
$$\boldsymbol{m}_i = \frac{1}{2}(1-s)(\boldsymbol{p}_{i+1} - \boldsymbol{p}_{i-1}). \tag{3.2}$$

Der Spannungparameter kontrolliert die Länge der Tangenten und somit die Krümmung der Kurve. Die Krümmung ist die Richtungsänderung pro Längeneinheit, die genaue Formel findet sich in Abschnitt 3.2.2. Für $s = -1$ ist die Krümmung minimal, für $s = 1$ maximal, der kardinale Spline degeneriert hier zu einem linearen Spline. Ein kardinaler Spline mit $s = 0,5$ wird in der Literatur oft auch als *Catmull-Rom Spline* bezeichnet.

Kochanek-Bartels Splines

Kardinale Splines erlauben nur eine globale Kontrolle des Kurvenverlaufs über den Spannungsparameter s. Kochanek-Bartels Splines [KB84] erlauben dagegen lokale Kontrolle über den Verlauf der Kurve. Jedem Punkt $\boldsymbol{p}_i$ sind drei Parameter zugeordnet: Ein Spannungsparameter $\boldsymbol{s}_i$, ein Verzerrungsparameter ($\boldsymbol{b}_i$) und ein Stetigkeitsparameter $\boldsymbol{c}_i$.

Der Spannungsparameter hat die gleiche Wirkung wie bei kardinalen Splines, der einzige Unterschied ist nur, dass nun jeder einzelne Punkt seinen eigenen Spannungsparameter hat. Mit dem Verzerrungsparameter wird gesteuert, wo der Wendepunkt liegt. Für $\boldsymbol{b}_i = 0$ liegt er im Punkt selbst, für $\boldsymbol{b}_i < 0$ vor und für $\boldsymbol{b}_i > 0$ nach dem Punkt.



(a) $s = 0, b = 0$ (b) $s = 1, b = 0$ (c) $s = -1, b = 0$ (d) $s = 0, b = 1$ (e) $s = 0, b = -1$

Abbildung 3.3: Spannung und Verzerrung bei Kochanek-Bartels Splines

Abbildung 3.3 illustriert die Wirkung der beiden Parameter. Sie zeigt fünf Kochanek-Bartels Splines durch drei Punkte $\boldsymbol{p}_0$, $\boldsymbol{p}_1$ und $\boldsymbol{p}_2$, die sich nur in der Wahl der Parameter $\boldsymbol{s}_1$ und $\boldsymbol{b}_1$ unterscheiden. Die gewählten Parameter sind direkt unter der jeweiligen Kurve vermerkt.

Der Stetigkeitsparameter wird $\boldsymbol{c}_i$ in dieser Arbeit nicht weiter betrachtet, da jeder andere Wert als 0 zwangsläufig eine Kurve liefert, die nicht C^1 stetig ist. Die genaue Formel zur Berechnung einer Tangenten $\boldsymbol{m}_i$ zu einem Punkt $\boldsymbol{p}_i$ ist:

$$\boldsymbol{m}_i = \frac{(1 - \boldsymbol{s}_i)(1 + \boldsymbol{b}_i)}{2}(\boldsymbol{p}_i - \boldsymbol{p}_{i-1}) + \frac{(1 - \boldsymbol{s}_i)(1 - \boldsymbol{b}_i)}{2}(\boldsymbol{p}_{i+1} - \boldsymbol{p}_i). \tag{3.3}$$

3.2 Heuristische Bestimmung des kombinatorischen Anteils eines MTSP

In diesem Abschnitt sollen Splines zur näherungsweisen Bestimmung des kombinatorischen Anteils, also der Reihenfolge, in der die Städte angefahren werden, eingesetzt werden. Die Zielsetzung ist die Minimierung der gesamten Fahrzeit. Die Frage ist nun, von welchen Faktoren die Fahrzeit abhängt.

Sie ist einerseits abhängig von den Eigenschaften des Fahrzeugs, die über das Differentialgleichungsssystem sowie die Beschleunigungs- und Geschwindigkeitsbeschränkungen bestimmt sind. Andererseits ist die Fahrzeit noch vom zweiten Punkt, den Eigenschaften der Bahn, abhängig.

3.2.1 Eigenschaften optimaler Bahnen

Bei gegebenem Fahrzeug ist die für eine Bahn / Kurve benötigte Fahrzeit von zwei Faktoren abhängig: Der *Bogenlänge* sowie der durchschnittlichen *Krümmung* der Kurve. Die beiden Begriffe müssen zunächst definiert werden [Wü97].

Die *Bogenlänge* $L(C_f)$ ist die Länge der Kurve $C_f = \{\boldsymbol{p} \in \mathbb{R}^n : \boldsymbol{p} = f(t),\ t \in [a, b]\}$. Die exakte Formel für die Berechnung ist

$$L(C_f) = \int_a^b \|f'(t)\| dt, \tag{3.4}$$

wobei $\|f'(t)\|$ die *euklidsche Länge* des Vektors $f'(t)$ bezeichnet. Für eine ebene Kurve mit $f(t) = \{x(t), y(t)\}$ und wird die Formel daher zu

$$L(C_f) = \int_a^b \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2} dt. \tag{3.5}$$

Die Formel ist analytisch integrierbar [BSM05], aus Gründen der Übersichtlichkeit wird jedoch auf eine Darstellung verzichtet. Sie lässt sich auch mit numerischen Verfahren, wie beispielsweise der *adaptiven Simpson Quadratur*, integrieren. Die MATLAB Funktion `quad` implementiert dieses Verfahren und wurde in dieser Arbeit für die Bestimmung der Bogenlänge verwendet.

Die *Krümmung* $\kappa(t)$ ist als Richtungsänderung der Kurve pro Längeneinheit definiert. Bei einem gegebenen physikalischen Modell und einer gegebenen parametrischen Kurve $f(t) = (x(t), y(t))$ bestimmt die lokale Krümmung

$$\kappa(t) = \left| \frac{\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)}{\left(\dot{x}(t)^2 + \dot{y}(t)^2\right)^{3/2}} \right| \tag{3.6}$$

die im Punkt t maximal erzielbare Geschwindigkeit. Dieser Zusammenhang lässt sich gut anhand einer Rennstrecke und einem Rennauto erklären.

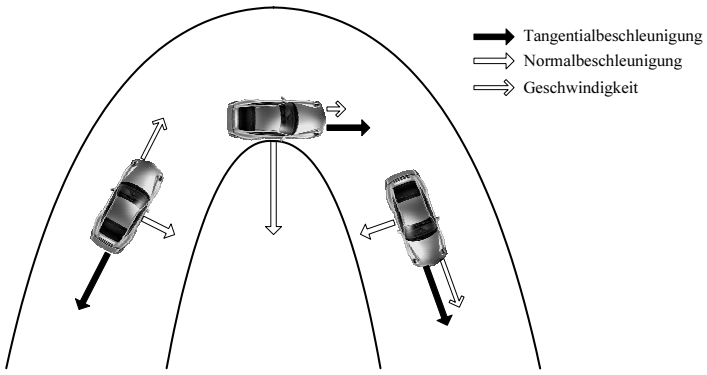


Abbildung 3.4: Verteilung der Beschleunigungsenergie

Abbildung 3.4 zeigt ein Rennauto in verschiedenen Phasen einer Kurvenfahrt. Sei die maximale Länge des Beschleunigungsvektors $\vec{a} = (a_x(t), a_y(t))$ durch $a_x(t)^2 + a_y(t)^2 \leq a_{limit}$ beschränkt. Der Beschleunigungsvektor lässt sich zu jedem Zeitpunkt der Fahrt durch einen Basistausch in zwei Komponenten transformieren: Eine Tangentialkomponente a_{tan} und eine Normalkomponente a_{norm} [HMS06]. Die Tangentialkomponente entspricht der Beschleunigung / Verzögerung des Rennwagens in

Fahrtrichtung, die Normalkomponente ist orthogonal dazu und entspricht der seitlichen Beschleunigung. Beide Komponenten sind in der Grafik durch verschiedene Pfeile dargestellt. Der dritte Pfeil visualisiert die (Tacho-)Geschwindigkeit des Fahrzeugs.

Das Beschleunigungslimit determiniert die maximal mögliche Geschwindigkeit in jedem Punkt auf der Kurve. Diese muss so gering sein, dass mit der verfügbaren Beschleunigungsenergie die Bahn gehalten werden kann. Andernfalls fliegt das Fahrzeug „aus der Kurve“. Die maximal mögliche Geschwindigkeit in einem Punkt wird im folgenden als *Grenzgeschwindigkeit* bezeichnet.

Die in Abbildung 3.4 dargestellte Fahrt durch ein Kurvensegment ist typisch: Je enger die Kurve bzw. je größer die Krümmung wird, desto mehr muss die Geschwindigkeit reduziert werden um die Bahn halten zu können. Ist der Punkt maximaler Krümmung des Segments, auch als *Scheitelpunkt* bezeichnet, passiert, so sinkt die Krümmung und die Geschwindigkeit kann wieder erhöht werden.

Wie muss nun eine Kurve durch eine Folge von Punkten bzw. Städten $\boldsymbol{p}_i,\ i = 1, ..n$ aussehen, damit die Fahrzeit für ein gegebenes Fahrzeug minimal ist? Eine erste Antwort ist: Die Kurve muss so gestaltet sein, dass möglichst viel der verfügbaren Beschleunigungsenergie in Geschwindigkeit umgesetzt werden kann.

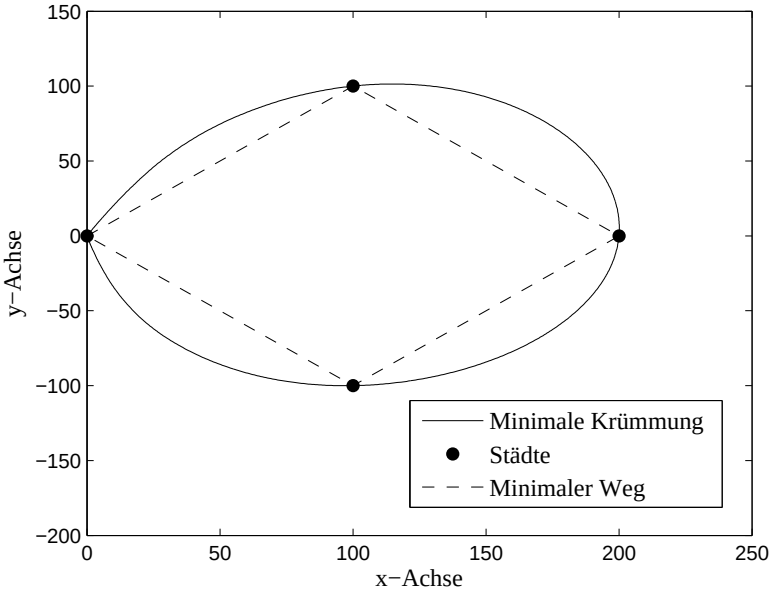


Abbildung 3.5: Vergleich zweier Kurven

Eine solche Kurve ist genau eine Kurve mit minimaler durchschnittlicher Krümmung. Dies lässt sich gut anhand des Vergleichs der Bahn einer optimalen MTSP Lösung mit der Lösung eines klassischen TSP demonstrieren. Abbildung 3.5 zeigt die beiden Kurven sowie die zu besuchenden Städte.

Ein Fahrzeug, dass die Lösung des klassischen TSP abfahren will, muss wie folgt vorgehen: Es beschleunigt auf jedem Segment in Tangentialrichtung bis zur Hälfte der Strecke, danach muss es wieder abbremsen, um in der nicht differenzierbaren

Ecke mit einer Geschwindigkeit von 0 zu drehen. Dieser Prozess wiederholt sich für die folgenden Segmente.

Auf der Kurve minimaler Krümmung sieht das Abfahren anders aus. Anfangs beschleunigt das Fahrzeug hauptsächlich in Tangentialrichtung, mit zunehmender Geschwindigkeit muss immer mehr Beschleunigungsenergie in Normalrichtung investiert werden, bis schließlich die Grenzgeschwindigkeit erreicht ist und nicht weiter in Tangentialrichtung beschleunigt werden kann. Auf der gesamten restlichen Strecke variiert die Geschwindigkeit nur noch minimal, fast die gesamte Energie wird in Richtungsänderungen, also in die Normalkomponenten a_{norm}, investiert. Am Ende der Fahrt wird die Geschwindigkeit kontinuierlich reduziert um mit $v = 0$ im Ursprung anzukommen.

Anders ausgedrückt bremst das Fahrzeug auf einer Kurve minimaler durchschnittlicher Krümmung so wenig wie möglich, folglich wird die Durchschnittsgeschwindigkeit maximiert.

Ist die Reihenfolge der Städte nicht gegeben, so spielt zusätzlich zu der Eigenschaft minimaler durchschnittlicher Krümmung noch die Bogenlänge eine Rolle. Im nächsten Abschnitt wird weiter auf diesen Punkt eingegangen.

3.2.2 Das Verfahren und Ergebnisse

Um ein MTSP optimal lösen zu können, muss für jedes Teilproblem im Branch & Bound Baum ein Optimalsteuerungsproblem gelöst werden. Jedes dieser Optimalsteuerungsprobleme lässt sich in zwei Teilprobleme zerlegen, die sequentiell gelöst werden können:

1. Bestimmung der optimalen Bahn
2. Bestimmung der optimalen Steuerung zum Abfahren dieser Trajektorie

Aus dieser Zerlegung wird ersichtlich, dass nur die Güte der Bahn den Zielfunktionswert t_f beeinflusst, da im zweiten Schritt lediglich die Steuerungen zum Abfahren dieser bestimmt werden.

Ist kein Geschwindigkeitslimit vorgesehen, so kann die optimale Bahn durch die Folge von Städten $\boldsymbol{p}_i,\ i = 1, ..., n$ durch die Lösung des folgenden Optimierungsproblems gefunden werden:

$$\text{Minimiere} \qquad \bar{\kappa} = \int_a^b \kappa(t)dt \tag{3.7}$$

unter der Nebenbedingung

$$f(t) = (x(t), y(t)) \text{ ist ein kubischer Hermite Spline durch } \boldsymbol{p}_1, ..., \boldsymbol{p}_n. \tag{3.8}$$

Die Minimierung der Fläche unter der Krümmungsfunktion entspricht der Minimierung der durchschnittlichen Krümmung. Das Problem wurde mit Hilfe des MATLAB

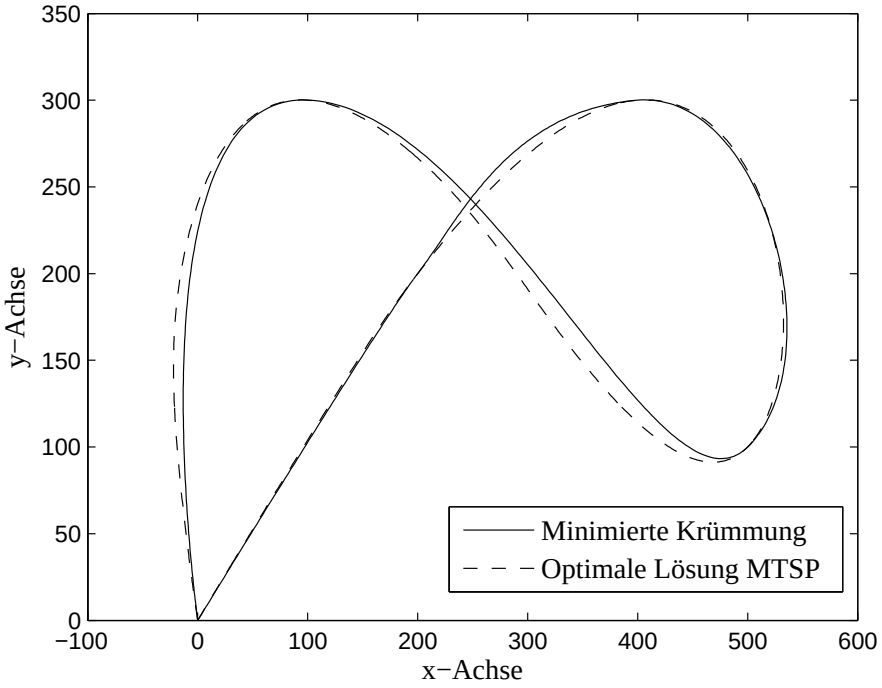


Abbildung 3.6: MTSP Bahn im Vergleich zu einem optimierten Spline

Solvers `fmincon` exemplarisch für eine Folge von Städten gelöst, die zu optimierenden Variablen waren die Tangenten $\boldsymbol{m}_i$ in den einzelnen Städten. Als Startlösung wurden die Tangenten eines kardinalen Splines mit $s = -0,4$ verwendet.

Abbildung 3.6 zeigt den optimalen Spline im Vergleich zu optimalen Bahn des Optimalsteuerungsproblems für eine bestimmte Folge von Städten. Wie man sieht, weichen die Kurven nur minimal voneinander ab.

Ist die Reihenfolge der Städte also vorgegeben, so lässt sich eine optimale Bahn bestimmen. Da jedoch die optimale Reihenfolge der Städte gefunden werden soll, müssen die optimalen Bahnen aller möglichen Permutationen der Städte miteinander verglichen werden. Dabei spielt nun nicht nur die durchschnittliche Krümmung der Kurven, sondern auch ihre jeweilige Bogenlänge eine Rolle.

Sei K eine Indexmenge, die alle möglichen Permutationen der Reihenfolge der Städte enthält und $C = \{C_k | k \in K\}$ die Menge der optimalen Bahnen. Eine Kurve $C_i \in C$ liefert die kürzeste Fahrzeit, wenn sie sowohl die kleinste durchschnittliche Krümmung $\bar{\kappa_i}$ als auch die kürzeste Bogenlänge $L(C_i)$ unter allen Kurven hat. Ist dies nicht der Fall, muss auf einen Qualitätsindex zurückgegriffen werden, der die Kurven anhand ihrer Krümmung und Bogenlänge bewertet. In der hier beschriebenen Heuristik wird dazu der Index $\Omega_i = \frac{1}{2}L(C_i) + \frac{1}{2}\bar{\kappa_i}$ verwendet. Je kleiner der Wert, desto höher die Qualität der Kurve.

Mit dem Index und dem Branch & Bound Verfahren von Dakin ließe sich nun eine Startschätzung für die optimale Reihenfolge der Städte ermitteln. Es wird eine weitere Vereinfachung vorgenommen, da die Ermittlung der optimalen Bahn für jede Permutation der Städte zuviel Zeit in Anspruch nehmen wurde. Statt einem auf minimales $\bar{\kappa}$ optimierten Spline kommt für jede Permutation ein einfacher kardinaler

Spline mit $s = -0,4$ zum Einsatz. Er liefert im Schnitt die besten Ω Werte unter allen möglichen Parameterwerten im Intervall $[-1, 1]$.

Formuliert man anhand der getroffenen Annahmen ein Optimierungsmodell, so erhält man das Nebenbedingungssystem eines klassisches TSP (1.2) - (1.6) zusammen mit einer anderen Zielfunktion. Statt einer Kostensumme wird nun der Ω-Wert der durch die Städte gelegten kardinalen Splines minimiert.

Das Branch & Bound Verfahren von Dakin kann praktisch unverändert auf diese neue Problemstellung angewendet werden, es ändert sich einzig und allein das Verfahren zur Bestimmung der „Kosten“ der aktuellen Tour.

Die so ermittelte Lösung sollte eine gute Startschätzung für den kombinatorischen Anteil eines MTSP darstellen, da sie aus den Eigenschaften optimaler Bahnen abgeleitet wurde. Sie ist auch für Probleme mit einer Geschwindigkeitsbeschränkung einsetzbar, da auch hier der optimale Weg eine Kurve minimaler durchschnittlicher Krümmung ist.

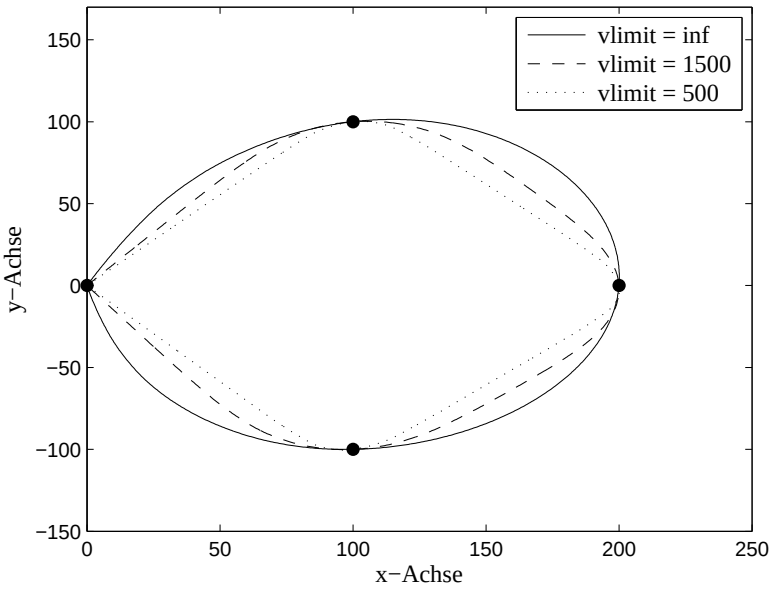


Abbildung 3.7: Verschiedene Geschwindigkeitslimits

Abbildung 3.7 zeigt die optimalen Trajektorien durch eine Menge von Städten bei verschiedenen Geschwindigkeitslimits. Sowohl im Fall von $v_{limit} = 1500$ als auch von $v_{limit} = 500$ limitiert das Geschwindigkeitslimit, und nicht das Beschleunigungslimit in Kombination mit der Krümmung der Kurve, die maximal erreichbare Geschwindigkeit zwischen zwei Städten.

Der Verlauf der Kurve verändert sich, da es keinen Sinn macht, einen längeren Weg mit einer höheren Grenzgeschwindigkeit zu wählen, wenn diese aufgrund des Geschwindkeitslimits sowieso nicht erreicht werden kann. Folglich wird ein kürzerer Weg gewählt, für den die Grenzgeschwindigkeit dem Geschwindigkeitslimit entspricht. Je mehr das Geschwindigkeitslimit die Geschwindigkeit begrenzt, desto mehr nähert sich der Weg einer Gerade an.

Städte	Durchschnitt	Median	Varianz
4	0,41	0,22	0,23
5	0,77	0,49	0,51
6	2,66	1,20	22,24
7	5,36	2,33	189,14

Tabelle 3.1: Rechenzeiten in Sekunden für die Bestimmung einer Startlösung

Zum Abschluss dieses Kapitels noch die Rechenzeiten für den Algorithmus. Dazu wurden je 100 Probleme mit 4 bis 7 Städten zufällig erzeugt und mit dem Branch & Bound Verfahren von Dakin sowie dem Index Ω als Kostenfunktion gelöst. Tabelle 3.1 zeigt die durchschnittliche Rechenzeit, den Median (50% Quantil) sowie die Varianz in Sekunden [*sec*].

Der Median liegt deutlich unterhalb der durchschnittlichen Rechenzeit, was daran liegt, dass es einige „Ausreißer“ nach oben gab. Ebenso sieht man an den Rechenzeiten die exponentielle Komplexität des Problems: Eine Erhöhung der Anzahl der Städte um 1 führt ca. zu einer Verdopplung der mittleren Rechenzeit.

3.3 Approximative Bestimmung von Fahrzeiten für gegebene Kurven

Bei dem Problem, zu einer gegebenen Kurve die Fahrzeit bestimmen, die ein gegebenes Fahrzeug zum Abfahren benötigt, handelt es sich um ein Optimalsteuerungsproblem. Die Zielsetzung ist es, sowohl den Bahnfehler (die Abweichung von der vorgegebenen Bahn), als auch die Fahrzeit zu minimieren.

Das Problem ist einfacher zu lösen, als die im Rahmen des Branch & Bound Verfahrens für MTSPs auftretenden Optimalsteuerungsprobleme, da nicht simultan die optimale Bahn bestimmt werden muss. Es ist dennoch zu komplex, um es im Rahmen eines Ansatzes zur schnellen Bestimmung unterer Schranken exakt zu lösen.

Aus diesem Grund wird in diesem Abschnitt versucht, auf Basis von Spline-Kurven gute Schätzungen für die Fahrzeiten zu bestimmen, die möglichst die reale Fahrzeit unterschätzen und folglich als untere Schranken im Rahmen des Branch & Bound Verfahrens eingesetzt werden können. Dazu werden zunächst die Unterschiede und Gemeinsamkeiten von optimalen Lösungen eines Optimalsteuerungsproblems und Spline-Kurven identifiziert sowie analysiert.

Eine offensichtliche Gemeinsamkeit ist, dass zweidimensionale Splines genau wie die Ortsfunktionen eines MTSP einen Weg im $\mathbb{R}^2$ beschreiben. Abbildung 3.8 zeigt die optimale Ortsfunktion eines MTSP im Vergleich zu einem kardinalen Spline durch die gleiche Reihenfolge der Städte. Es fällt auf, dass sich die beiden Kurven relativ ähnlich sehen. Was die Abbildung dagegen nicht zeigt, sind die völlig unterschiedlichen Bedeutungen des Parameters t bei Ortsfunktionen und Splines.

Bei den beiden Ortsfunktionen $x(t)$ und $y(t)$ eines MTSP entspricht der Parameter $t \in [0, t_f]$ der *Realzeit* in Sekunden. Startet das Fahrzeug zum Zeitpunkt 0, so liefern $x(t)$ und $y(t)$ also die x- bzw. y-Position des Fahrzeugs nach t Sekunden Fahrzeit. Analog liefern die Funktionen $v_x(t)$, $v_y(t)$, $a_x(t)$ und $a_y(t)$ die Geschwindigkeiten und Beschleunigungen nach t Sekunden Fahrzeit.

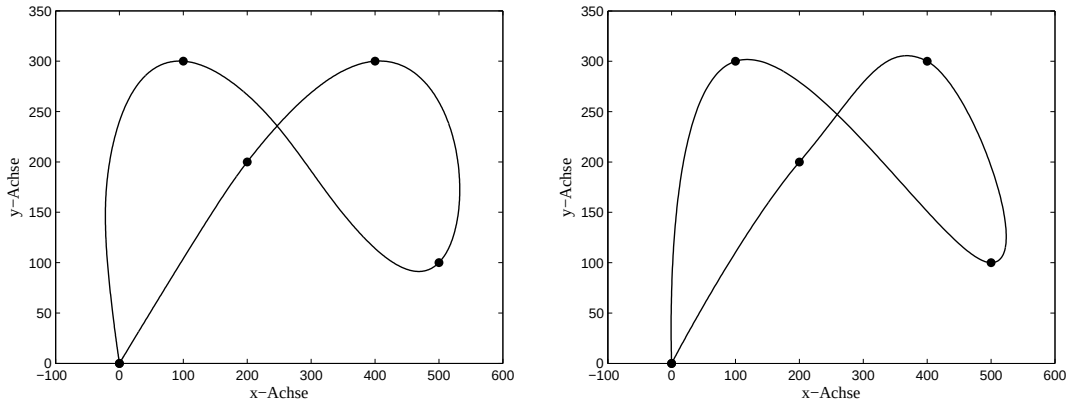
Bei einer Spline-Kurve hat $t \in [a, b]$ zunächst keinerlei zeitliche Bedeutung. Ein spezifisches t beschreibt lediglich einen Punkt auf der Kurve, a und b können beliebig gewählt werden. Eine sinnvolle Parametrisierung für einen Spline mit n Segmenten ist jedoch $a = 0$ und $b = n$, da *ein einzelnes Segment* bei den meisten Splines im Intervall $[0, 1]$ parametrisiert ist.

Interpretiert man eine Spline-Kurve jedoch als physikalische Ortsfunktion, so bekommt der Parameter t eine zeitliche Bedeutung. Als Basis für die folgenden Überlegungen dient dabei ein kubischer Hermite Spline mit n Segmenten $S_i,\ i = 1, .., n$. Jedes Segment wird an m Positionen $\boldsymbol{z}_j,\ j = 1, ..., m$ im Intervall $[0, 1]$ ausgewertet. Dabei gilt $\boldsymbol{z}_0 = 0$ und $\boldsymbol{z}_{j+1} = \boldsymbol{z}_j + d$ mit $d = \frac{1}{m}$. Zusammen mit dem Endpunkt des letzten Segments ergeben sich also insgesamt $(n \cdot m) + 1$ Punkte $\boldsymbol{p}_k,\ k = 1, ..., (n \cdot m) + 1$ auf der Kurve.

Zusätzlich werden auch noch die ersten und zweiten Ableitungen des Splines an jedem Punkt $\boldsymbol{p}_k$ bestimmt. Dazu werden analytisch die beiden Ableitungen $\boldsymbol{p}'(t)$ und $\boldsymbol{p}''(t)$ der Spline-Funktion (3.1) gebildet und an den gleichen Positionen wie die Spline-Funktion selbst ausgewertet.

Die Punkte liegen auf einem *uniformen parametrischen Raster*, der *parametrische Abstand* $|t_k - t_{k+1}|$ zwischen jedem Paar $(\boldsymbol{p}_k, \boldsymbol{p}_{k+1})$ von Punkten ist also gleich groß. Der parametrische Abstand ist der Abstand bezogen auf den Parameter t, er ist nicht mit dem geometrischen Abstand $\|p_{k+1} - p_k\|$ zu verwechseln.

Die grundlegende Idee ist es nun, zu den $(n \cdot m)+1$ Punkten auf der Kurve ein uniform unterteiltes Intervall $[0, t_f]$ zu finden, für das alle Geschwindigkeits- und Beschleunigungsbeschränkungen des physikalischen Fahrzeugmodells erfüllt sind. Dazu müssen zunächst die Eigenschaften einer uniformen Skalierung der Kurve untersucht werden. Sei dazu die Kurve auf dem Intervall $[0, t_f^0]$ definiert. Es lässt sich nun leicht



(a) MTSP (b) Kardinaler Spline ($s = -0, 4$)

Abbildung 3.8: Eine Ortsfunktion und die korrespondierende Spline-Kurve

überprüfen, ob die ersten und zweiten Ableitungen der Kurve die Beschränkungen in jedem Punkt erfüllen.

Tun sie dies, so ist die Kurve mit der Endzeit t_f^0 eine zulässige Lösung für das Optimalsteuerungsproblem. Wie wird aber vorgegangen, wenn die Beschränkungen nicht erfüllt sind? In diesem Fall muss eine Endzeit $t_f > t_f^0$ gewählt werden. Bevor auf das Verfahren zur Bestimmung eines solchen t_f eingegangen wird, müssen aber zunächst die Eigenschaften einer uniformen Skalierung von t_f^0 bestimmt werden.

Eine uniforme Skalierung des Endzeitpunkts t_f^0 mit dem Skalierungsfaktor α entspricht mathematisch der Substitution des Parameters t der Ortsfunktion $x(t)$ durch den Parameter $z = \frac{t}{\alpha},\ z \in [0, \alpha \cdot t_f]$. Die Funktion wird also in horizontaler Richtung gestaucht bzw. gestreckt. Anschließend werden die neuen ersten und zweiten Ableitungen sowie die Funktionen $a_x(t)^2 + a_y(t)^2$ und $v_x(t)^2 + v_y(t)^2$ berechnet.

Liegen ein oder mehrere Punkte über den vorgegebenen Limits, so muss t_f kontinuierlich erhöht werden, andernfalls kann der Parameter so lange verringert werden, bis ein Punkt genau das Limit erreicht. Es ergibt sich ein einfaches Optimierungsmodell mit der einzigen Variablen t_f:

$$\text{Minimiere } t_f \tag{3.9}$$

unter den Nebenbedingungen

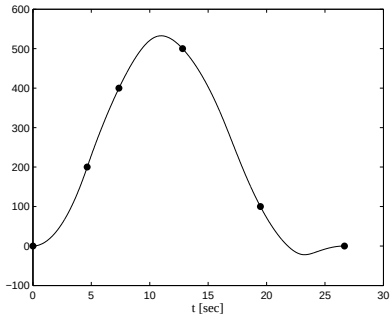
$$\max(a_x(t)^2 + a_y(t)^2) \leq a_{limit}, \quad t \in [0, t_f] \tag{3.10}$$
$$\max(v_x(t)^2 + v_y(t)^2) \leq v_{limit}, \quad t \in [0, t_f] \tag{3.11}$$
$$t_f \geq 0. \tag{3.12}$$

Das Optimierungsmodell lässt sich in ein konvexes Nullstellenproblem transformieren, welches sich leicht mit Standardverfahren der unbeschränkten nichtlinearen Optimierung lösen lässt. In dieser Arbeit wurde dazu die Funktion `fminsearch` aus der Optimization Toolbox von MATLAB verwendet.

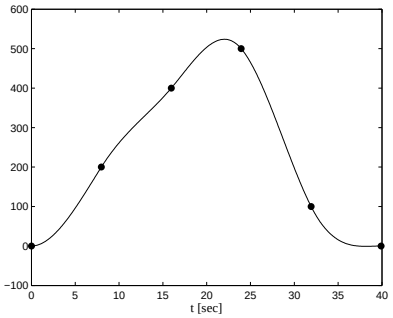
3.3.1 Analyse der Ergebnisse

Die so bestimmten Endzeitpunkte t_f liegen jedoch weit höher als die bei der optimalen Lösung eines MTSP erzielten. Dies liegt einerseits daran, dass die Physik bei der Konstruktion der Spline-Kurve nicht vollständig berücksichtigt wird, andererseits aber auch daran, dass ein uniformes Raster verwendet wird. Die genaue Problematik lässt sich am besten anhand eines Beispiels verdeutlichen. Die dazu nötigen Grafiken zeigen die Abbildungen 3.9 und 3.10.

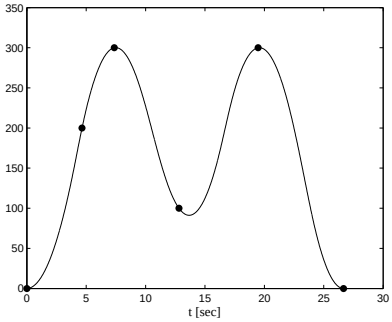
Betrachtet wurde ein Optimalsteuerungsproblem ohne Geschwindigkeitslimitmit, aber mit einem Beschleunigungslimit von $a_x(t)^2 + a_y(t)^2 \leq 700$. Durch die bekannte Reihenfolge der Städte wurde anschließend ein kardinaler Spline mit Spannungsparameter $s = -0,4$ gelegt und die dazugehörige Fahrzeit durch die Lösung von



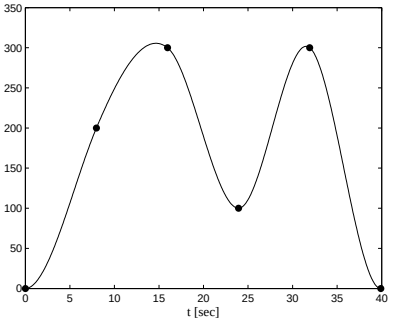
(a) $x(t)$ Referenzlösung MTSP



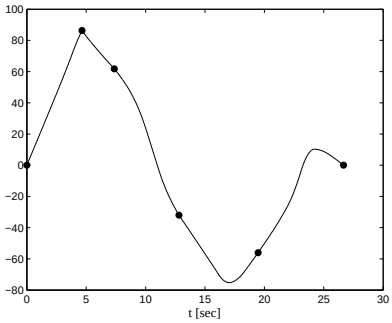
(b) $x(t)$ Spline



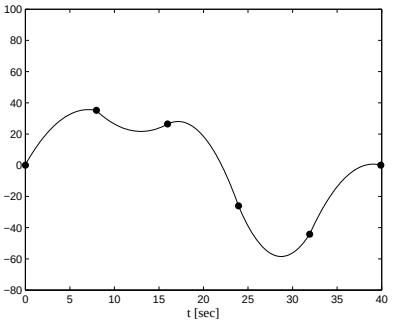
(c) $y(t)$ Referenzlösung MTSP



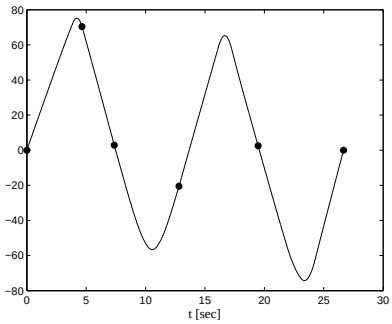
(d) $y(t)$ Spline



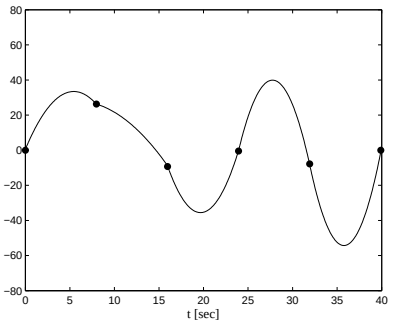
(e) $v_x(t)$ Referenzlösung MTSP



(f) $v_x(t)$ Spline

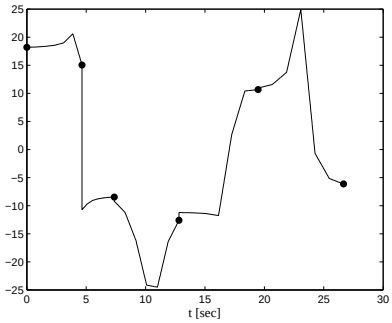


(g) $v_y(t)$ Referenzlösung MTSP

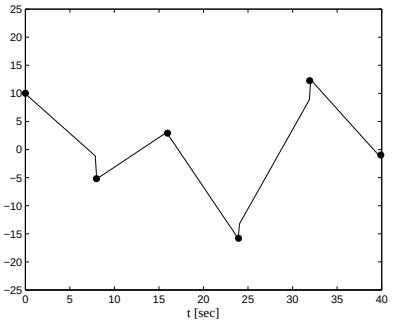


(h) $v_y(t)$ Spline

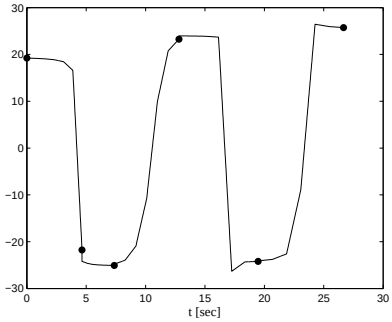
Abbildung 3.9: Vergleich der Kurvenverläufe I



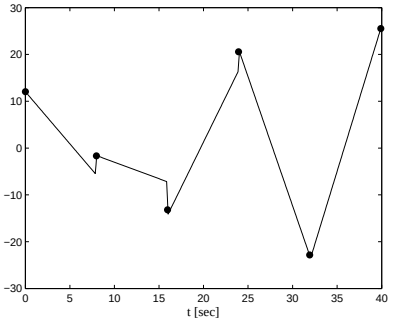
(a) $a_x(t)$ Referenzlösung MTSP



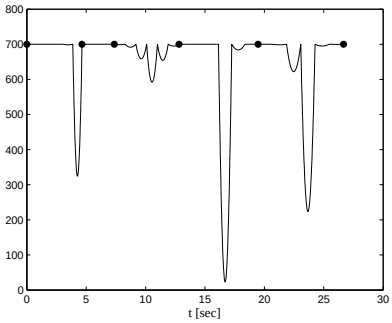
(b) $a_x(t)$ Spline



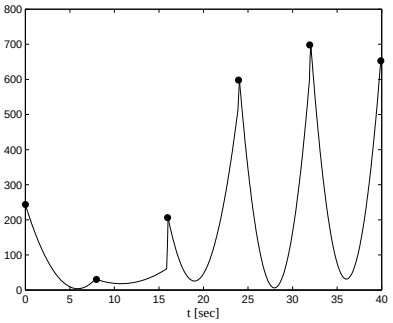
(c) $a_y(t)$ Referenzlösung MTSP



(d) $a_y(t)$ Spline



(e) $a_x(t)^2 + a_y(t)^2$ Referenzlösung MTSP



(f) $a_x(t)^2 + a_y(t)^2$ Spline

Abbildung 3.10: Vergleich der Kurvenverläufe II

(3.9)-(3.12) bestimmt. Die Fahrzeit der optimalen Lösung beträgt 27 sec, die Fahrzeit für den kardinalen Spline dagegen 40 sec. Woher kommt dieser Unterschied von fast 50%?

Vergleicht man die beiden Ortsfunktionen $x(t)$ und $y(t)$ des kardinalen Splines, so fällt auf, dass diese vor Erreichen des Maximums von $x(t)$ deutlich flacher verlaufen als die korrespondierenden Abschnitte der Ortsfunktionen des MTSP. Die Geschwindigkeit des Fahrzeugs ist in diesem Abschnitt folglich deutlich geringer als bei der optimalen Lösung des MTSP, der Verlauf der Geschwindigkeiten $v_x(t)$ und $v_y(t)$ bestätigt dies.

Eine Erklärung dafür liefert die Betrachtung des Beschleunigungslimits $a_x(t)^2 + a_y(t)^2$. Die Kurve liegt in dem betrachteten Abschnitt weit unter der Grenze von 700, das Fahrzeug ist also langsamer, weil es seine Beschleunigungskapazitäten nicht vollständig ausnutzt.

Ganz anders dagegen der Verlauf des Beschleunigungslimits bei der optimalen Lösung des MTSP. Hier nutzt das Fahrzeug fast über den gesamten Zeitraum seine maximal verfügbare Beschleunigungsenergie aus, zusammen mit der optimalen Bahn ergibt sich eine minimale Fahrzeit.

Bei gegebener Strecke ist die Ausnutzung der verfügbaren Beschleunigungsenergie des Fahrzeugs eine notwendige Bedingung für eine minimale Fahrzeit. Damit der kardinale Spline aber eine primal zulässige Lösung für das Steuerungsproblem liefert, muss er *in jedem Punkt* die Beschleunigungsbeschränkung erfüllen. Betrachtet man nochmal die entsprechende Grafik des kardinalen Spline, so fällt auf, dass die Kurve einen sehr ungünstigen Verlauf hat. Der Endzeitpunkt t_f wird einzig und allein durch das weit über dem Durchschnitt liegende Maximum dieser Funktion determiniert.

Dazu kommt, dass bei einem uniformen Raster im Intervall $[0, t_f]$ die Variation des Endzeitpunkts t_f nur die Amplituden, aber nicht den Verlauf von $a_x(t)^2 + a_y(t)^2$ verändert. In der ursprünglichen Parametrisierung war der parametrische Abstand $|t_i - t_{i+1}|$ zwischen zwei aufeinander folgenden Städten $\boldsymbol{p}_i$ und $\boldsymbol{p}_{i+1},\ i = 1, ..., n-1$ immer 1. Es gilt folglich $|t_{i-1} - t_i| = |t_i - t_{i+1}|,\ \forall i$.

Skaliert man die Zeitachse zur Interpretation der Parametrisierung als Realzeit, so werden die Abstände zwischen den einzelnen Paaren von Städten zwar größer bzw. kleiner, sie bleiben aber alle gleich groß. In anderen Worten benötigt das Fahrzeug immer die gleiche Zeit, um von einer Stadt zur nächsten zu gelangen, egal wie die Städte zueinander liegen.

Betrachtet man die zeitlichen Abstände für die Fahrt von einer Stadt zur nächsten in Referenzlösung des MTSP, so fällt auf, dass die benötigten Zeiten für die Fahrt von einer Stadt zu nächsten variieren. Dies entspricht auch der Intuition, schließlich macht es in der Realität sehr wohl einen Unterschied, ob z.B. die nächste Stadt nach Darmstadt nun Weiterstadt oder Moskau ist.

Bei der auf diesem Weg ermittelten Lösung bestimmt das „schwierigste“ Segment die Gesamtfahrtzeit, die ermittelte Fahrzeit kann folglich als obere Schranke für die reale Fahrzeit auf der Kurve betrachtet werden. Die ermittelten Zustände und Steuerungen können als Startlösung für ein im Rahmen des B&B Verfahrens für MTSPe zu lösendes NLP verwendet werden.

Um eine bessere Approximation für die Fahrzeit zu ermitteln, muss aber ein anderer Ansatz entwickelt werden. Dieser baut auf der hier dargestellten Skalierung der Zeitachse auf und wird im folgenden Abschnitt dargestellt.

3.3.2 Eine bessere Approximation der Fahrzeit

Die möglichst vollständige Ausnutzung der verfügbaren Beschleunigungsenergie ist, wie im letzten Abschnitt beschrieben, eine notwendige Bedingung für die Erzielung einer optimalen Fahrzeit auf einer gegebenen Kurve. Eine mögliche mathematische Formulierung für diese Bedingung ist

$$\text{mean}(a_x(t)^2 + a_y(t)^2) = \max(a_x(t)^2 + a_y(t)^2) = a_{limit},\ t \in [0, t_f]. \tag{3.13}$$

Die einzigen Funktionen, für die der Durchschnitt ($mean$) auf einem definierten Intervall gleich dem Maximum (max) ist, sind konstante Funktionen. Diese Formulierung wurde jedoch bewusst so gewählt, da sich aus ihr ein Verfahren ableiten lässt, mit dem sich möglicherweise eine untere Schranke für die Fahrzeit durch eine bekannte Folge von Städten bestimmen lässt.

Es sei ein physikalisches Modell ohne Geschwindigkeitsbeschränkung gegeben. Wie im vorherigen Abschnitt wird wieder die Zeitachse skaliert, allerdings wird jetzt gefordert, dass der Durchschnitt und nicht das Maximum der Funktion $a_x(t)^2+a_y(t)^2$ dem Beschleunigungslimit a_{limit} entspricht. Dadurch geht die primale Zulässigkeit der ermittelten Lösung verloren, dies ist aber bei jedem Verfahren zur Bestimmung einer unteren Schranke zwangsläufig der Fall.

Das zu (3.9)-(3.12) sehr ähnliche Optimierungsmodell lautet:

$$\text{Minimiere } t_f \tag{3.14}$$

unter den Nebenbedingungen

$$\int_0^{t_f} (a_x(t)^2 + a_y(t)^2)\, dt = a_{limit} \cdot t_f \tag{3.15}$$

$$t_f \geq 0. \tag{3.16}$$

Auch dieses Problem lässt sich mit `fminsearch` sehr schnell (ca. 1 ms) lösen. Um die Tauglichkeit der so ermittelten Fahrzeit als untere Schranke zu beurteilen, wurden zwei MTSP Instanzen betrachtet. Für jede Instanz wurde die optimale Fahrzeit für eine gegebene Reihenfolge von Städten mit der optimalen Lösung von (3.14)-(3.16) verglichen.

Städte	1	2	3	4
x-Koordinate	200	400	500	100
y-Koordinate	200	300	100	300

Tabelle 3.2: Daten der ersten MTSP Instanz

Tabelle 3.2 zeigt die zu besuchenden Städte der ersten MTSP Instanz. Die Beschleunigungsbeschränkung lag bei $a_{limit} = 900$, der verwendete Spline war ein kardinaler Spline mit $s = -0,4$. Tabelle 3.3 zeigt zu jeder Besuchsreihenfolge die jeweils optimalen Lösungen und die mit dem Verfahren ermittelten Fahrzeiten.

Reihenfolge	1-2-3-4	1-2-4-3	2-3-1-4	1-3-2-4	1-3-4-2
t [sec] Optimal	25,04	30,53	25,65	23,19	31,97
t [sec] Modell	25,23	30,5	26,42	22,77	30,84

Tabelle 3.3: Fahrzeiten für gegebene Besuchsreihenfolgen

Wie man an der Tabelle sieht, liegt die ermittelte Zeit nur bei einer der untersuchten Reihenfolgen leicht oberhalb der optimalen Fahrzeit. Das Problem ist, dass dies nicht bei allen Probleminstanzen der Fall ist und auch keine Aussage darüber getroffen werden kann, unter welchen Umständen es funktioniert und unter welchen nicht.

Tabelle 3.4 zeigt exemplarisch eine Menge von Städten, für die das Modell sehr schlechte Ergebnisse liefert, die fast alle oberhalb der optimalen Fahrzeit liegen. Das Beschleunigungslimit lag in diesem Fall bei $a_{limit} = 1000$.

Städte	1	2	3	4	5
x-Koordinate	70	20	45	80	300
y-Koordinate	400	100	55	10	100

Tabelle 3.4: Daten der zweiten MTSP Instanz

In Tabelle 3.5 sind zu einer Besuchsreihenfolge die jeweils optimalen Zeiten in Sekunden [*sec*] sowie dem Modell ermittelten Fahrzeiten dargestellt.

Reihenfolge	1-2-3-4-5	2-1-5-4-3	3-2-4-5-1	4-3-2-1-5
t [sec] Optimal	26,02	23,08	23,82	22,04
t [sec] Modell	30,47	24,74	28,29	27,08

Tabelle 3.5: Fahrzeiten für gegebene Besuchsreihenfolgen II

Wie man leicht sieht, gelingt es dem Modell für keine der untersuchten Besuchsreihenfolgen, die tatsächliche Fahrzeit zu approximieren, aber nicht zu unterschätzen. Das Modell ist zur Berechnung einer unteren Schranke folglich nicht geeignet. In den beiden nächsten Abschnitten wird zunächst eine echte untere Schranke für Probleme mit Geschwindigkeitslimit und danach für Probleme ohne Geschwindigkeitslimit vorgestellt.

3.4 Eine untere Schranke für Probleme mit Geschwindigkeitslimit

Im Falle eines gegebenen Geschwindigkeitslimits fällt die Bestimmung einer unteren Schranke für die Fahrzeit leichter als im Fall ohne Limit. Sei dazu eine Reihenfolge von n Städten $\boldsymbol{p}_i,\ i = 1, ..., n$ gegeben. Zudem wird o.B.d.A. angenommen, dass der Index i die Besuchsreihenfolge angibt, also Stadt i direkt vor Stadt $i + 1$ besucht wird.

Zunächst wird aus dem Geschwindigkeitslimit eine obere Schranke $\overline{v}$ für die Durchschnittsgeschwindigkeit bestimmt. Eine solche obere Schranke ist die maximale Länge des Geschwindigkeitsvektors $\vec{v} = (v_x(t), v_y(t))$. Diese lässt sich aus dem Geschwindigkeitslimit $v_x(t)^2 + v_y(t)^2 \leq v_{limit}$ ableiten. Dazu geht man davon aus, dass das Fahrzeug sich zu einem Zeitpunkt t mit maximaler Geschwindigkeit in x-Richtung bewegt, es gilt also $v_x(t)^2 = v_{limit}$ und folglich $\overline{v} = \max \|\vec{v}\| = \sqrt{v_{limit}}$.

Als nächstes wird eine untere Schranke $\underline{L}(C_n)$ für die Länge $L(C_n)$ der Kurve C_n durch die n Städte bestimmt werden. Die kürzeste Kurve ergibt sich, wenn man alle Paare von Städten $(\boldsymbol{p}_i, \boldsymbol{p}_{i+1})$ über die Luftlinie verbindet. Folglich gilt

$$\underline{L}(C_n) = \sum_{i=1}^{n-1} |\boldsymbol{p}_i - \boldsymbol{p}_{i+1}|. \tag{3.17}$$

Mit diesen zwei Größen lässt sich nun eine untere Schranke $\underline{t}_f$ für die Fahrzeit t_f berechnen:

$$\underline{t}_f = \frac{\underline{L}(C_n)}{\overline{v}}. \tag{3.18}$$

Dass es sich bei $\underline{t}_f$ um eine untere Schranke handelt, folgt direkt daraus, dass $\underline{L}(C_n)$ eine untere Schranke für die Weglänge und $\overline{v}$ eine obere Schranke für die Durchschnittsgeschwindigkeit des Fahrzeugs ist. Ein Fahrzeug mit dem Geschwindigkeitslimit v_{limit} kann also eine gegebene Sequenz von Städten $\boldsymbol{p}_1, ..., \boldsymbol{p}_n$ nicht in weniger als $\underline{t}_f$ Sekunden besuchen.

Wie scharf $\underline{t}_f$ ist, also wie nah der Wert am optimalen t_f^* liegt, ist abhängig vom Verhältnis des Geschwindigkeitslimits v_{limit} zum Beschleunigungslimit a_{limit} sowie der unteren Schranke $\underline{L}(C_n)$ für die Weglänge. Betrachtet man nochmals Abbildung 3.7, so wird klar wieso. Je kleiner das Geschwindigkeitslimit im Vergleich zum Beschleunigungslimit wird, desto mehr nähert sich die optimale Kurve den direkten Wegen an und desto länger fährt das Fahrzeug mit Maximalgeschwindigkeit.

Tabelle 3.6 zeigt die optimale Fahrzeit t_f^*, sowie die Werte der unteren Schranke $\underline{t}_f$ und einer verbesserten unteren Schranke $\underline{t}_f^+$ für verschiedene Kombinationen von Beschleunigungs- und Geschwindigkeitsbeschränkungen. Alle Zeiten sind in Sekunden $[sec]$ angegeben.

Beschleunigungslimit	1000	1000	1000	1000	1000
Geschschwindigkeitslimit	2000	1500	1000	500	300
Optimale Fahrzeit t_f^*	14,82	16,48	19,41	26,42	33,71
Untere Schranke $\underline{t}'_f$	12,65	14,61	17,89	25,30	32,66
Verbesserte Schranke $\underline{t}_f^+$	14,06	15,83	18,89	26,01	33,21

Tabelle 3.6: Vergleich unterer Schranken mit der optimalen Lösung

Wie man sieht, ist die untere Schranke $\underline{t}_f$ vor allem bei hohen Geschwindigkeitslimits schlecht. Für $v_{limit} = 2000$ liegt der Wert der Schranke 15% unter dem Optimum t_f^*, bei $v_{limit} = 300$ sind es dagegen nur 3%. Eine der Ursachen für diesen Unterschied ist, dass die untere Schranke nicht berücksichtigt, wie lange das Fahrzeug am Anfang der Fahrt zum Erreichen der Maximalgeschwindigkeit und am Ende zum Abbremsen auf 0 benötigt.

Die verbesserte Schranke $\underline{t}_f^+$ berücksichtigt diese Zeiten, dadurch sinkt die Abweichung bei $v_{limit} = 2000$ auf 5% und bei $v_{limit} = 300$ auf 2%. Da es sich bei dem betrachteten Problem um ein sehr einfaches handelt, ist für kompliziertere Probleme eine höhere Abweichung zu erwarten als für das betrachtete.

Sei $\overline{a}$ die maximale Länge des Beschleunigungsvektors bei gegebenem a_{limit}, die Berechnung erfolgt analog zu $\overline{v}$. Mit dieser Größe und der physikalischen Formel für eine gleichförmig beschleunigte Bewegung lässt sich nun eine untere Schranke für die Zeit t_a angeben, die das Fahrzeug zum Erreichen der maximalen Geschwindigkeit $\overline{v}$ benötigt. Es gilt $t_a = \overline{v}/\overline{a}$, da das physikalische Gesetz $\vec{v} = \vec{a} \cdot t$ lautet.

Anschließend wird die Länge des Weges L_a berechnet, den das Fahrzeug zum Beschleunigen benötigt. Es gilt unter der Annahme einer gleichförmig beschleunigten Bewegung $L_a = \frac{1}{2} \cdot \overline{a} \cdot t_a^2$. Den gleichen Weg und die gleiche Zeit benötigt das Fahrzeug auch am Ende der Fahrt, um wieder auf eine Geschwindigkeit von 0 abzubremsen.

Sei $\underline{L}'(c_n) = \underline{L}(c_n) - 2 \cdot L_a$ der verbleibende Weg, auf dem sich das Fahrzeug mit maximaler Geschwindigkeit bewegen kann. Die verbesserte Schranke lässt sich nun wie folgt berechnen:

$$\underline{t}_f^+ = 2 \cdot t_a + \frac{\underline{L}'(c_n)}{\overline{v}}. \tag{3.19}$$

3.5 Eine untere Schranke für Probleme ohne Geschwindigkeitslimit

Aufbauend auf der verbesserten unteren Schranke $\underline{t}_f^+$ für Probleme mit Geschwindigkeitslimit lässt sich auch eine beweisbare untere Schranke $\underline{t}'_f$ für Probleme ohne Geschwindigkeitslimit konstruieren.

Es wird angenommen, dass das Fahrzeug auch hier den kürzesten möglichen Weg $\underline{L}(C_n)$ zurücklegen muss. Dazu wird angenommen, dass das Fahrzeug seine gesamte Beschleunigungsenergie in die Tangentialbeschleunigung a_{tan} investieren kann. Dies entspricht einer Probleminstanz, bei der alle Städte auf einer Linie liegen.

Das Fahrzeug kann folglich bis zur Hälfte der Strecke $\underline{L}(C_n)$ mit der maximalen Tangentialbeschleunigung beschleunigen, die zweite Hälfte der Strecke muss es mit der maximalen Beschleunigungsenergie bremsen, um im Ursprung wieder zum Stillstand zu kommen. Die exakte Formel zur Berechnung dieser Schranke ist

$$\underline{t}'_f = 2 \cdot \sqrt{\frac{\underline{L}(C_n)}{\overline{a}}}. \tag{3.20}$$

Die Beweisidee ist folgende: Liegen alle Städte auf einer Linie, so kann das Fahrzeug seine gesamte Beschleunigungsenergie in Tangentialgeschwindigkeit umsetzen. Da es keinen kürzeren, alle Städte verbindenden Weg als $\underline{L}(C_n)$ gibt, kann das Fahrzeug folglich nicht weniger als $\underline{t}'_f$ Sekunden brauchen, um alle Städte zu besuchen.

Reihenfolge	1-2-3-4	1-2-4-3	2-3-1-4	1-3-2-4	1-3-4-2
Optimale Fahrzeit t_f^*	25,04	30,53	25,65	23,19	31,97
Untere Schranke $\underline{t}'_f$	14,11	15,33	14,13	13,85	15,69

Tabelle 3.7: Vergleich unterer Schranken mit der optimalen Lösung II

Tabelle 3.7 vergleicht die Werte der unteren Schranke $\underline{t}'_f$ mit der jeweils optimalen Lösung t_f^* für das bereits in Abschnitt 3.3.2 beschriebene Beispielproblem. Die gerade eben beschriebene Schranke ist nicht scharf, sie unterschätzt die optimalen Zeiten t_f^* viel zu stark.

Der Grund dafür ist, dass das Fahrzeug einen Teil der verfügbaren Beschleunigungsenergie in die Normalbeschleunigung a_{norm} stecken muss, um Kurven fahren zu können. Dies wird bei der Berechnung der Schranke allerdings komplett ignoriert, folglich weichen die Ergebnisse stark vom Optimum ab. Die Schranke ist in einem Branch & Bound Verfahren also von keinem großen Wert, da sich nur sehr wenige Teilprobleme durch sie ausloten lassen werden.

Eine Verbesserung der Schranke durch Berücksichtigung dieses Umstandes ist leider nicht einfach möglich. Die Lage der Städte zueinander entscheidet, wie viel der verfügbaren Energie auf einer Fahrt in die Tangential- und wie viel in die Normalbeschleunigung investiert werden muss. Diese Verteilung lässt sich aber nur durch Lösung des Optimalsteuerungproblems exakt bestimmen.

Kapitel 4

Physikalische Spline-Kurven

In diesem Kapitel wird nun ein weiterer Ansatz zur Approximation der Fahrzeit für Probleme ohne Geschwindigkeitslimit vorgestellt. Im Gegensatz zu dem Modell aus Abschnitt 3.3 wird die Physik hier jedoch bereits bei der Konstruktion der Kurven explizit berücksichtigt. Es ergeben sich physikalisch motivierte Spline-Kurven.

Die Grundlage für diese physikalischen Spline-Kurven bildet das Gesetz der *gleichförmig beschleunigten Bewegung*. Es beschreibt die Bewegung eines Massepunkts, auf den eine konstante Kraft bzw. Beschleunigung a wirkt. Das klassische Beispiel für eine solche Bewegung ist der freie Fall mit $a = g = 9.81 \frac{m}{sec^2}$.

Die im Zeitraum $[0, t]$ zurückgelegte Strecke $s(t)$ eines gleichförmig mit a beschleunigten Massepunkts lässt sich durch die folgende Formel beschreiben [HMS06]:

$$s(t) = \frac{1}{2} \cdot a \cdot t^2. \tag{4.1}$$

Die dabei erreichte Endgeschwindigkeit $v(t)$ ist

$$v(t) = \dot{s}(t) = a \cdot t. \tag{4.2}$$

Die Formeln lassen sich noch allgemeiner formulieren. Sei $\boldsymbol{p}_0 \in \mathbb{R}^n$ die Anfangsposition des Massepunkts in einem n-dimensionalen Vektorraum, $\boldsymbol{v}_0 \in \mathbb{R}^n$ der Vektor der Anfangsgeschwindigkeit in jeder Dimension und $\boldsymbol{a} \in \mathbb{R}^n$ der Vektor der konstanten Beschleunigungen in jede Dimension. Dann lässt sich seine Position $\boldsymbol{p}(t) \in \mathbb{R}^n$ nach t Zeiteinheiten durch

$$\boldsymbol{p}(t) = \frac{1}{2} \cdot \boldsymbol{a} \cdot t^2 + \boldsymbol{v}_0 \cdot t + \boldsymbol{p}_0. \tag{4.3}$$

bestimmen. Der Vektor der Endgeschwindigkeiten ist nun $\boldsymbol{v}(t)$ ist

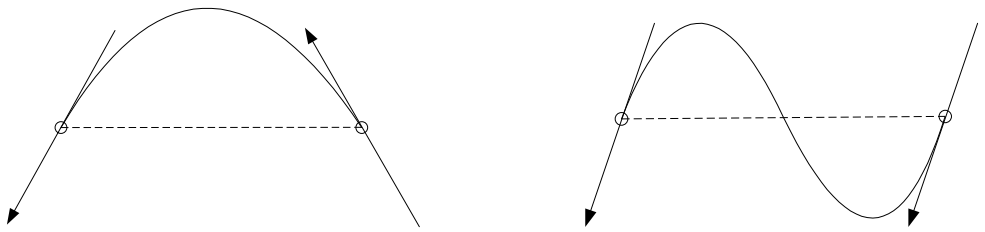
$$\boldsymbol{v}(t) = \dot{\boldsymbol{p}}(t) = \boldsymbol{a} \cdot t + \boldsymbol{v}_0. \tag{4.4}$$

Aus Gleichung (4.3) lässt sich nun ein n-dimensionaler, *quadratischer Spline* konstruieren. Bevor im Folgenden auf die Konstruktion der Kurve eingegangen wird, müssen jedoch noch einige Vorüberlegungen zu quadratischen Kurven angestellt werden.

4.1 Quadratische Kurven

Die grundlegende Frage ist, ob die Verwendung von quadratischen statt kubischen Segmenten gravierende Einschränkungen mit sich bringt. Bezüglich der vom MTSP geforderten C^1 Stetigkeit stellen quadratische Segmente (Polynomgrad = 3) keine Einschränkung dar, da sich aus ihnen nach Abschnitt 3.1.1 C^1 stetige Kurven konstruieren lassen.

Als nächstes ist die Frage zu klären, ob quadratische Segmente den möglichen Kurvenverlauf einschränken. Sie lässt sich am besten anhand eines kleinen Beispiels beantworten. Abbildung 4.1 zeigt zwei Mal das gleiche Kurvensegment, jedoch mit unterschiedlichen Tangenten in den Kontrollpunkten.



(a) realisierbar (b) nicht realisierbar

Abbildung 4.1: Mit quadratischen Segmenten realisierbare Verläufe

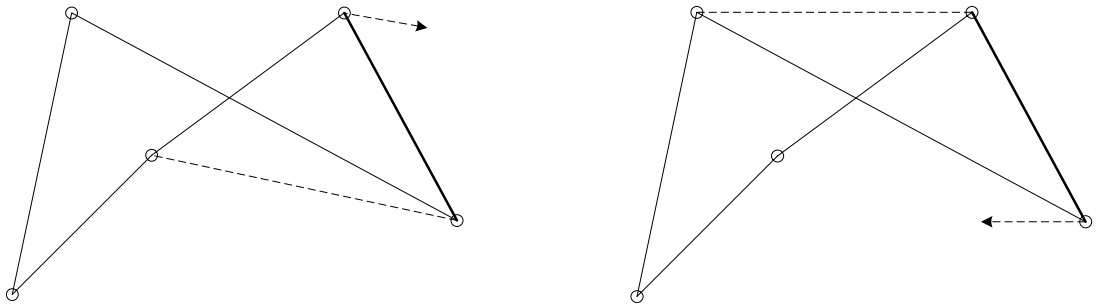
Während der links abgebildete Kurvenverlauf sowohl mit einer kubischen als auch einer quadratischen Basisfunktion realisierbar ist, ist für den rechts abgebildeten Verlauf eine kubische Basisfunktion erforderlich. Der Grund ist, dass für diesen Kurvenverlauf ein Wendepunkt, also eine Nullstelle in der zweiten Ableitung, erforderlich ist. Da die zweite Ableitung einer quadratischen Funktion jedoch eine Konstante ist, kann kein Wendepunkt existieren.

Man überlegt sich jedoch leicht, dass sich diese Limitation durch das Einführen eines expliziten Wendepunkts in der Mitte zwischen den beiden Kontrollpunkten näherungsweise auflösen lässt. Mann muss also überprüfen, wann ein kubisches Segment erforderlich ist und es dann durch die Einführung eines zusätzlichen Kontrollpunkts in zwei quadratische Segmente aufsplitten. Die Frage ist nun, wie man eine solche Situation erkennt.

Betrachtet man die Start- und Endtangenten beider Segmente in Abbildung 4.1, so sieht man leicht, wann ein Wendepunkt und somit ein kubisches Segment erforderlich ist. Dies ist nämlich genau dann der Fall, wenn beide Tangenten des Segments *in die gleiche Richtung* von der gestrichelt dargestellten Verbindungslinie zwischen den Kontrollpunkten weg zeigen.

Nun stellen sich zwei praktische Fragen: Wie bestimmt man aus einer gegebenen Reihenfolge von Kontrollpunkten (Städten) die Tangenten in den Kontrollpunkten und wie lässt sich einfach bestimmen, ob beide Tangenten in die gleiche Richtung zeigen?

Die Tangente in einem Kontrollpunkt $\boldsymbol{p}_i$ bestimmt man, wie bei kardinalen Splines, anhand der umliegenden Kontrollunkte $\boldsymbol{p}_{i-1}$ und $\boldsymbol{p}_{i+1}$. Der Vektor $\boldsymbol{m}_i = \boldsymbol{p}_{i+1} - \boldsymbol{p}_{i-1}$ ist die Tangente. Die Länge ist egal, da hier nur die Richtung interessiert. Abbildung 4.2 zeigt das Verfahren für das fett markierte Segment graphisch. Da die Tangenten in verschiedene Richtungen zeigen, muss kein expliziter Wendepunkt eingeführt werden.

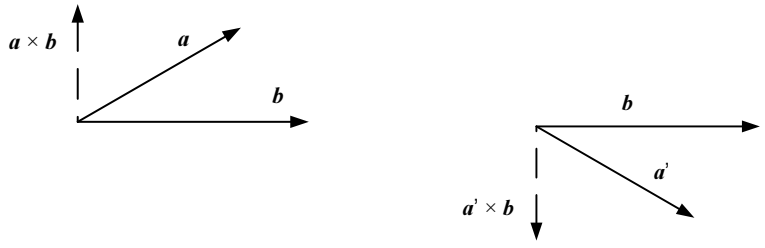


(a) Starttangente (b) Endtangente

Abbildung 4.2: Start- und Endttangenten

Ob beide Tangenten eines Segments in die selbe Richtung zeigen, lässt sich für den hier gegebenen zweidimensionalen Fall mit Hilfe des *Kreuzprodukts* [Wü97] $\boldsymbol{c} = \boldsymbol{a} \times \boldsymbol{b}$, $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}^3$ feststellen. Der Ergebnisvektor $\boldsymbol{c}$ ist orthogonal zu den beiden Vektoren $\boldsymbol{a}$ und $\boldsymbol{b}$, er steht also senkrecht auf der von beiden aufgespannten Ebene. Seine Länge entspricht der Fläche des von $\boldsymbol{a}$ und $\boldsymbol{b}$ aufgespannten Parallelogramms.

Es kommen jedoch zwei Vektoren gleicher Länge, aber mit entgegengesetzten Richtungen in Frage, die senkrecht auf der Ebene stehen. Welcher der beiden Vektoren der korrekte ist, ist abhängig von der Lage von $\boldsymbol{a}$ relativ zu $\boldsymbol{b}$. Schaut man in der von $\boldsymbol{a}$ und $\boldsymbol{b}$ aufgespannten Ebene in Richtung des Vektors $\boldsymbol{b}$, so kann $\boldsymbol{a}$ nach links oder nach rechts von $\boldsymbol{b}$ weg zeigen. Zeigt $\boldsymbol{a}$ nach rechts, so ist $\boldsymbol{c}$ positiv, zeigt $\boldsymbol{a}$ nach links, so ist $\boldsymbol{c}$ negativ. In Abbildung 4.3 ist dieser Zusammenhang noch einmal graphisch dargestellt.



(a) $\boldsymbol{a}$ links von $\boldsymbol{b}$ (b) $\boldsymbol{a}'$ rechts von $\boldsymbol{b}$

Abbildung 4.3: Das Kreuzprodukt

Sei das aktuelle Segment durch den Startpunkt $\boldsymbol{p}_i$ und den Endpunkt $\boldsymbol{p}_{i+1}$ definiert, dazu seien $\boldsymbol{m}_i$ sowie $\boldsymbol{m}_{i+1}$ die korrespondierenden Tangenten. Der Vektor $\boldsymbol{p} = \boldsymbol{p}_{i+1} - \boldsymbol{p}_i$ ist die Verbindung zwischen $\boldsymbol{p}_i$ und $\boldsymbol{p}_{i+1}$ und zeigt in Richtung $\boldsymbol{p}_{i+1}$. Um das

Kreuzprodukt anwendbar zu machen, werden alle Vektoren um eine z-Komponente mit dem Wert 0 erweitert.

Als erstes werden die beiden Kreuzprodukte $cross_1 = \boldsymbol{p} \times \boldsymbol{m}_i$ sowie $cross_2 = \boldsymbol{p} \times \boldsymbol{m}_{i+1}$ berechnet. Danach muss überprüft werden, in welche Richtung die beiden Vektoren zeigen. Da sich alle Eingabevektoren ausschließlich in der xy-Ebene befinden, ist es ausreichend, nur das Vorzeichen der z-Komponente der beiden Kreuzprodukte zu überprüfen.

Sind die beiden Vorzeichen gleich, so zeigen $\boldsymbol{m}_i$ und $\boldsymbol{m}_{i+1}$ in die gleiche Richtung von $\boldsymbol{p}$ weg. Es wird intuitiv ein zusätzlicher, zwischen $\boldsymbol{p}_i$ und $\boldsymbol{p}_{i+1}$ liegender, Kontrollpunkt mit den Koordinaten $\boldsymbol{p}_{new} = \frac{1}{2}\boldsymbol{p}_i + \frac{1}{2}\boldsymbol{p}_{i+1}$ eingefügt.

4.2 Konstruktion physikalischer Splines

Nach der notwendigen Erweiterung der Kontrollpunkte kann nun die eigentliche Konstruktion der Spline-Kurve beschrieben werden. Das Ziel ist eine C^1 stetige Kurve, bei der der Parameter t der Realzeit entspricht. Als Basis für jedes Segment dient Gleichung 4.3.

Die Konstruktion dieser physikalisch motivierten Kurve ist, im Gegensatz zur Konstruktion der Splines aus Kapitel 3.1, ein Optimierungsmodell mit den folgenden Parametern

$\boldsymbol{p}_i$	Kontrollpunkte der Kurve
a_{limit}	Beschleunigungslimit des Fahrzeugs
v_{limit}	Geschwindigkeitslimit des Fahrzeugs

und den folgenden Variablen

$\boldsymbol{a}_i$	Beschleunigungsvektor zwischen $\boldsymbol{p}_i$ und $\boldsymbol{p}_{i+1}$
$\boldsymbol{v}_i$	Geschwindigkeitsvektor in Punkt $\boldsymbol{p}_i$
t_i	Zeitpunkt, zu dem Punkt $\boldsymbol{p}_i$ erreicht wird.

Ein Modell mit n Kontrollpunkten hat insgesamt $m = 5 \cdot n$ Variablen, da jeder der Vektoren a_i und v_i je zwei skalare Variablen enthält. Das Modell wird zunächst verbal erläutert, darauf folgt eine mathematische Formulierung.

Die Zielfunktion des Modells ist identisch zu der des Optimalsteuerungsproblems: Zu minimieren ist der Zeitpunkt, zu dem das Fahrzeug wieder im Ursprung ankommt. Der Ursprung ist sowohl der erste als auch der letzte Kontrollpunkt der Kurve, folglich muss die Variable t_n minimiert werden.

Die Nebenbedingungen lassen sich in drei Gruppen einteilen. Die erste Gruppe umfasst dabei Regularitätsbedingungen für die Zeiten t_i, die zweite Gruppe C^0 und

C^1 Stetigkeitsbedingungen. Die dritte Gruppe umfasst schließlich Geschwindigkeits- und Beschleunigungsbeschränkungen.

Da das Modell möglichst eine untere Schranke für die optimale Lösung der im B&B Baum des MTSP zu lösenden Optimalsteuerungsprobleme liefern soll, wird auf die Randbedingungen verzichtet. Man erlaubt also, dass das Fahrzeug im ersten Kontrollpunkt $\boldsymbol{p}_1$ mit einer Anfangsgeschwindigkeit $\boldsymbol{v}_1 > 0$ startet und den letzten Punkt $\boldsymbol{p}_n$ mit einer Geschwindigkeit $\boldsymbol{v}_n > 0$ passiert.

Die mathematische Formulierung des Modells ist:

$$\text{Minimiere } t_n \tag{4.5}$$

unter den Nebenbedingungen

$$t_1 = 0 \tag{4.6}$$

$$t_i \leq t_{i+1} \qquad \forall i = 1, ..., n-1 \tag{4.7}$$

$$\boldsymbol{p}_{i+1} = \frac{1}{2} \cdot \boldsymbol{a}_i \cdot (t_{i+1} - t_i)^2 + \boldsymbol{v}_i \cdot (t_{i+1} - t_i) + \boldsymbol{p}_i \qquad \forall i = 1, ..., n-1 \tag{4.8}$$

$$\boldsymbol{v}_{i+1} = \boldsymbol{a}_i \cdot (t_{i+1} - t_i) + \boldsymbol{v}_i \qquad \forall i = 1, ..., n-1 \tag{4.9}$$

$$a_{limit} \geq \boldsymbol{a}_i^T \cdot \boldsymbol{a}_i \qquad \forall i = 1, ..., n \tag{4.10}$$

$$v_{limit} \geq \boldsymbol{v}_i^T \cdot \boldsymbol{v}_i \qquad \forall i = 1, ..., n \tag{4.11}$$

Die Nebenbedingung (4.6) fordert, dass das Fahrzeug zum Zeitpunkt 0 in $\boldsymbol{p}_1$ startet. Bedingung (4.7) fordert darauf aufbauend, dass die Werte der t_i mit steigendem i monoton wachsen. Das Fahrzeug muss also den auf $\boldsymbol{p}_i$ folgenden Kontrollpunkt $\boldsymbol{p}_{i+1}$ zu einem späteren Zeitpunkt als $\boldsymbol{p}_i$ erreichen. Zusammen bilden diese Nebenbedingungen die Regularitätsanforderungen an die Zeiten t_i.

Der nahtlose Anschluss der einzelnen Segmente, also die C^0 Stetigkeit der Kurve, wird über die Nebenbedingungen (4.8) sicher gestellt. Die C^1 Stetigkeit, also der stetige Anschluss der ersten Ableitungen, wird von den Bedingungen (4.9) garantiert.

Die in Vektorschreibweise dargestellten Nebenbedingungsgruppen (4.10) und (4.11) fordern schließlich die Einhaltung der Beschleunigungs- und Geschwindigkeitsbeschränkungen des Fahrzeugs. Insgesamt hat das Modell somit $7 \cdot n - 3$ Nebenbedingungen. Existiert keine Geschwindkeitsbeschränkung, so werden die Nebenbedingungen (4.11) weg gelassen, die Zahl der Bedingungen reduziert sich auf $6 \cdot n - 4$.

Es handelt sich bei dem Modell um ein nichtlineares Optimierungsproblem, das mit dem in Kapitel 2.2.3 beschriebenen SQP-Verfahren (lokal) optimal gelöst werden kann. Es wurde im Rahmen dieser Arbeit sowohl mit dem SQP Solver aus der Optimization Toolbox von MATLAB (`fmincon`) als auch dem SNOPT Solver gelöst. Im nächsten Abschnitt werden nun die beiden Solver verglichen.

4.3 Lösung des Problems mit SNOPT und fmincon

Die Problemstruktur des vorliegenden Optimierungsproblems ist der eines aus direkter Kollokation entstandenen NLPs sehr ähnlich, die Jacobi-Matrizen sind auch hier dünn besetzt und strukturiert. Der SNOPT Solver ist auf genau solche Jacobi-Matrizen optimiert, während `fmincon` immer von vollbesetzten Jacobi-Matrizen ausgeht. Dazu funktioniert SNOPT um so besser, je weniger Freiheitsgrade bzw. je mehr aktive Nebenbedingungen ein Problem hat.

Diese Ausrichtung äußert sich auch in der Eingabesyntax der beiden Solver. Während bei SNOPT nur die von 0 verschiedenen Elemente der Jacobi-Matrix spezifiziert werden müssen, erwartet `fmincon` immer die Spezifikation der gesamten Matrix.

Generell können beide Solver auch ohne Spezifikation der Jacobi-Matrix nichtlineare Optimierungsprobleme lösen. Dies erfordert aber deutlich mehr Rechenzeit, da die Gradienten der Zielfunktion und der Nebenbedingungen durch finite Differenzen approximiert werden müssen. Es müssen also multiple Funktionsauswertungen durchgeführt werden, wo sonst nur eine Auswertung der Gradientenfunktion nötig ist.

Vergleicht man die Leistung der beiden Solver mit ihren Standard-Einstellungen bezüglich der Lösungsgüte, so lassen sich keine signifikanten Unterschiede feststellen. Dass die ermittelten Zielfunktionswerte sich um bis zu einen Prozentpunkt unterscheiden, dürfte ausschließlich auf unterschiedlich scharfe Standard-Abbruchkriterien zurückzuführen sein.

(a) t in *sec* ohne Geschwindigkeitlimit

	fmincon	SNOPT
Finite Differenzen	2,07	7,37
Jacobi-Matrix	0,76	0,86

(b) t in *sec* mit Geschwindigkeitlimit

	fmincon	SNOPT
Finite Differenzen	1,46	0,97
Jacobi-Matrix	0,68	0,36

Tabelle 4.1: Vergleich der Solver für ein Problem mit vier Städten

Tabelle 4.1 zeigt exemplarisch die Rechenzeiten in Sekunden [*sec*] der beiden Solver für ein Problem mit vier Städten. Zusammen mit Start- und Endpunkt sowie einem automatisch eingefügten Wendepunkt ergibt sich ein zu lösendes Optimierungsproblem mit $n = 7$ Kontrollpunkten. Die Rechnungen wurden auf einer Intel Core 2 Duo CPU mit 3 Gigahertz Taktfrequenz durchgeführt.

Betrachtet man die Situation ohne einzuhaltendes Geschwindigkeitslimit, so fällt auf, dass `fmincon` hier SNOPT sowohl bei der Verwendung von finiten Differenzen als auch bei spezifizierter Jacobi-Matrix überlegen ist. Besonders die Gradientenapproximation mittels finiter Differenzen in Kombination mit vielen Freiheitsgraden wirkt sich stark negativ auf die von SNOPT benötigte Rechenzeit aus.

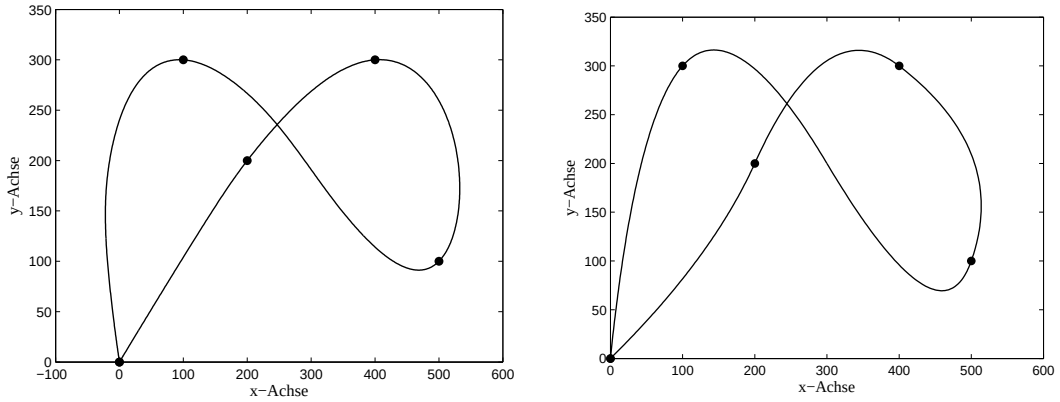
Ein aktives Geschwindigkeitslimit erhöht die Anzahl der Nebenbedingungen und senkt somit die Anzahl der Freiheitsgerade. Dies wirkt sich positiv auf die Rechenzeit beider Solver aus, allerdings profitiert SNOPT deutlich stärker davon als `fmincon` und kann diesen sowohl bei der Verwendung finiter Differenzen als auch bei spezifizierter Jacobi-Matrix überholen.

Dazu ist der SNOPT Solver für das untersuchte Problem stabiler als `fmincon`. Während SNOPT alle getesteten Probleme in innerhalb der standardmässig vorgegebenen Iterations- und Funktionsauswertungslimits lösen konnte, gab es bei `fmincon` Ausreißer, bei denen die Optimierung nicht innerhalb der Limits terminierte. Ob andere Startschätzungen als die verwendete ($a_i = v_i = (1,1)^T \; \forall i$, $t_i = i - 1 \; \forall i$) diese Ausreißer verhindern können, wurde nicht untersucht.

4.4 Analyse der Kurven

Was sind die Unterschiede zwischen bzw. die Gemeinsamkeiten von physikalisch motivierten Spline-Kurven und den optimalen Lösungen der Optimalsteuerungsprobleme? Um dieser Frage auf den Grund gehen zu können, werden im Folgenden die Verläufe der Ortsfunktionen $x(t)$ und $y(t)$, der Geschwindigkeitsprofile $v_x(t)$ und $v_y(t)$, der Beschleunigungsprofile $a_x(t)$ und $a_y(t)$ sowie die Beschleunigungslimits $a_x(t)^2 + a_y(t)^2$ gegenübergestellt.

Abbildung 4.4 zeigt den Verlauf der beiden Ortsfunktionen für das aus dem letzten Kapitel bekannte Beispielproblem ohne Geschwindigkeitsbeschränkung. Die Abbildungen 4.5 und 4.6 zeigen die Verläufe der restlichen Kurven.

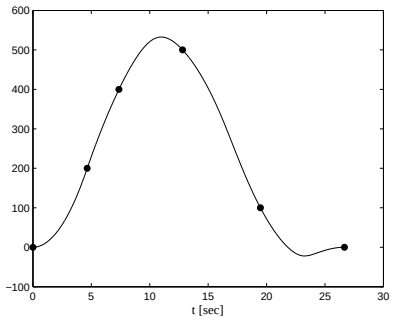


(a) MTSP (b) Physikalischer Spline

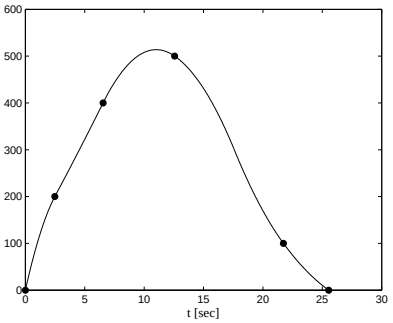
Abbildung 4.4: Eine Ortsfunktion und der korrespondierende physikalische Spline

Der Hauptunterschied ist, dass die beiden Kurven $v_x(t)$ und $v_y(t)$ nicht mit $v_x(0) = v_y(0) = 0$ starten und nicht mit $v_x(t_f) = v_y(t_f) = 0$ enden, sondern sich an diesem Punkten weit im positiven bzw. negativen Bereich befinden. Dementsprechend benötigt das Fahrzeug vom Ursprung bis zur ersten Stadt und von der letzten Stadt zurück zum Ursprung deutlich weniger Zeit als bei der Referenzlösung.

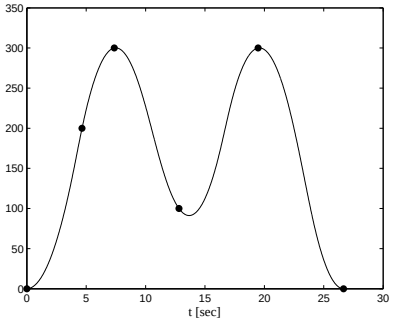
Betrachtet man den Verlauf der Beschleunigungsfunktionen $a_x(t)$ und $a_y(t)$, so sieht man direkt die eingeschränkte Steuerung: Für die gesamte Dauer der Fahrt von einer Stadt zur nächsten liegt ein konstanter Beschleunigungsvektor $\boldsymbol{a}_i$ an. Man sieht auch sehr gut, dass zwischen der dritten Stadt bei $(500, 100)$ und der vierten Stadt bei $(100, 300)$ ein zusätzlicher Wendepunkt eingefügt werden musste, um die



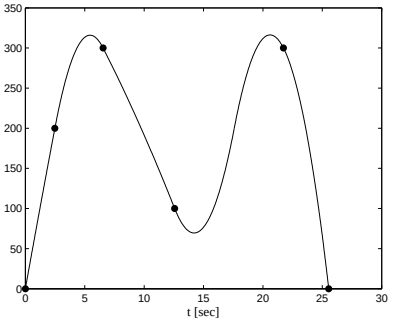
(a) $x(t)$ MTSP



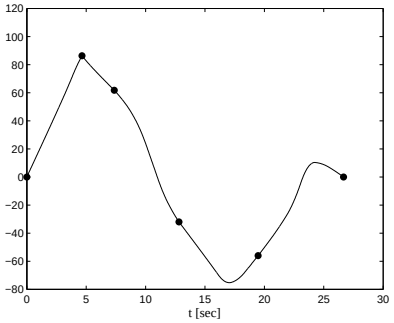
(b) $x(t)$ physikalischer Spline



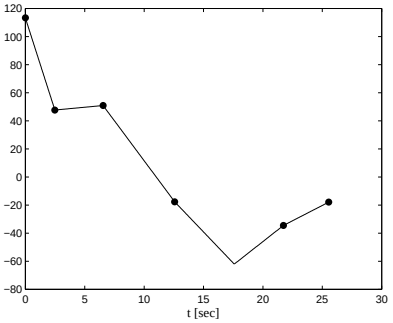
(c) $y(t)$ MTSP



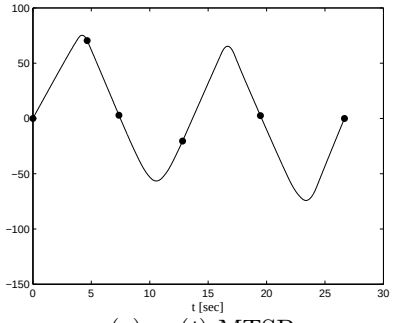
(d) $y(t)$ physikalischer Spline



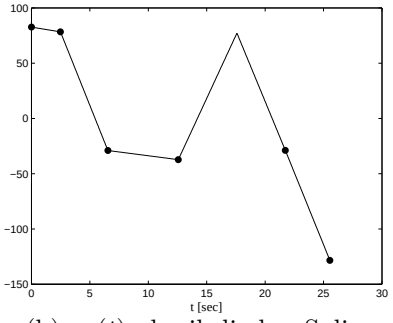
(e) $v_x(t)$ MTSP



(f) $v_x(t)$ physikalischer Spline

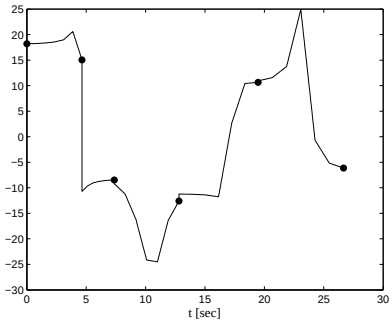


(g) $v_y(t)$ MTSP

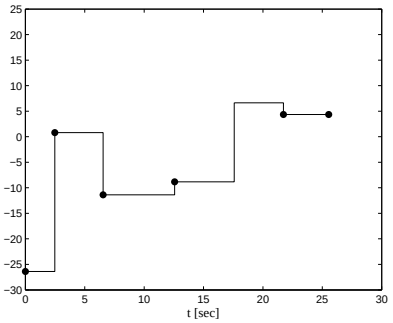


(h) $v_y(t)$ physikalischer Spline

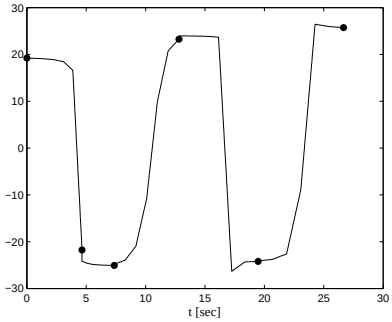
Abbildung 4.5: Vergleich der Kurvenverläufe MTSP und physikalischer Spline I



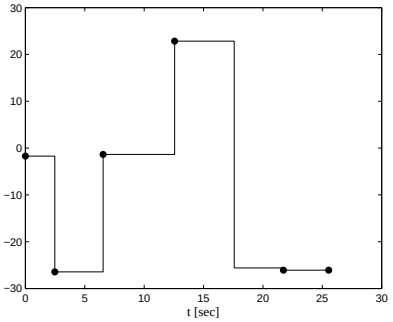
(a) $a_x(t)$ MTSP



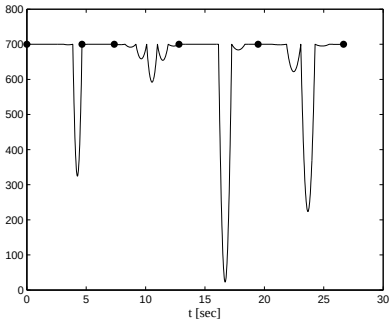
(b) $a_x(t)$ physikalischer Spline



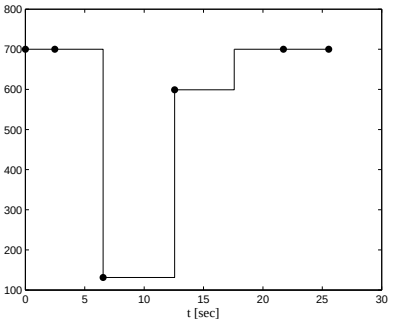
(c) $a_y(t)$ MTSP



(d) $a_y(t)$ physikalischer Spline



(e) $a_x(t)^2 + a_y(t)^2$ MTSP



(f) $a_x(t)^2 + a_y(t)^2$ phys. Spline

Abbildung 4.6: Vergleich der Kurvenverläufe MTSP und physikalischer Spline II

Limitationen eines quadratischen Splines zu umgehen. Zwischen diesen Städten ist die Steuerung nicht konstant.

Anhand der Beschleunigungsbeschränkung $a_x(t)^2 + a_y(t)^2$ sieht man gut, dass die eingeschränkte Steuerung auch dazu führt, dass die zu Verfügung stehende Beschleunigungsenergie nicht immer voll ausgenutzt werden kann. Im nächsten Abschnitt wird nun untersucht, inwiefern die Vorteile durch die relaxierten Randbedingungen die Nachteile durch die eingeschränkte Steuerung aufheben können.

4.5 Physikalische Splines als untere Schranken

In diesem Abschnitt wird die Tauglichkeit der physikalisch motivierten Spline Kurven als untere Schranken für ein im B&B Baum eines MTSP enthaltenes Optimalsteuerungsproblem untersucht. Es wird dazu die gleiche erste Probleminstanz wie in Kapitel 3.3.2 verwendet.

Reihenfolge	1-2-3-4	1-2-4-3	2-3-1-4	1-3-2-4	1-3-4-2
t [sec] Optimal	25,04	30,53	25,65	23,19	31,97
t [sec] Modell Kap. 3.3	25,23	30,5	26,42	22,77	30,84
t [sec] phys. Spline	23,98	30,21	24	20,31	32,31

Tabelle 4.2: Vergleich der beiden Verfahren I

Tabelle 4.2 zeigt die optimale Fahrzeit, die Werte der potentiellen Schranke aus Kapitel 3.3.2 sowie die Fahrzeiten für den physikalischen Spline. Die Zeiten des physikalischen Splines sind ähnlich zu denen, die mit dem Verfahren aus Kapitel 3.3.2 ermittelt wurden. Sie liegen für alle Besuchsreihenfolgen außer 1-3-4-2 unterhalb der optimalen Lösung. Nimmt man das Minimum aus den mit beiden Verfahren ermittelten Werten, so liegen alle Werte unterhalb der jeweils optimalen Lösung.

Von großem Interesse ist, wie sich der physikalische Spline im Vergleich zu dem Verfahren aus Kapitel 3.3.2 bei dem in Tabelle 3.4 gegebenen Problem verhält. Tabelle 4.3 zeigt die Ergebnisse.

Reihenfolge	1-2-3-4-5	2-1-5-4-3	3-2-4-5-1	4-3-2-1-5
t [sec] Optimal	26,02	23,08	23,82	22,04
t [sec] Modell Kap. 3.3	30,47	24,74	28,29	27,08
t [sec] phys. Spline	25,39	39,16	25,24	29,66

Tabelle 4.3: Vergleich der beiden Verfahren II

Bei diesem Problem verhält sich der physikalische Spline nicht besser als das Modell aus Kapitel 3.3.2. Allerdings lässt sich für den Spline eine qualitative Aussage treffen, wann er gut und wann weniger gut funktioniert. Dazu wird ein weiteres Problem mit 6 Städten betrachtet, auch hier wurde das Beschleunigungslimit auf $a_{limit} = 1000$ gesetzt.

Städte	1	2	3	4	5	6
x-Koordinate	475	300	445	228	410	307
y-Koordinate	115	240	380	10	220	395

Reihenfolge	1-5-2-6-3-4	2-6-3-5-1-4	3-6-2-5-1-4	4-5-3-6-2-1	5-3-6-2-1-4	6-3-1-5-2-4
t [sec] Optimal	27,49	25,35	26,76	27,11	26,25	27,80
t [sec] Modell Kap. 3.3	31,80	27,35	32,85	31,67	28,85	30,62
t [sec] phys. Spline	26,43	24,57	23,00	24,61	22,14	26,90

Tabelle 4.4: Daten einer Probleminstanz mit 6 Städten

Tabelle 4.4 zeigt sowohl die zu besuchenden Städte, als auch die Ergebnisse der Verfahren. Dabei fällt auf, dass alle Städte im Vergleich zum letzten Problem weit vom

Ursprung entfernt liegen. Dies eine Situation, die dem physikalischen Spline entgegenkommt, da hier die Vorteile der relaxierten Randbedingungen gut zur Geltung kommen.

Die angegebenen Ergebnisse bestätigen diese Vermutung. Während das Modell auf Kapitel 3.3.2 auch hier keine vernünftigen Ergebnisse liefert, kommen die Ergebnisse des Spline den optimalen Zeiten sehr nahe und unterschätzen diese für alle untersuchten Reihenfolgen.

Abschließend sei noch gesagt, dass der physikalische Spline auch auf Probleme mit Geschwindigkeitslimit anwendbar ist. Die Nachteile der groben Steuerung treten hier aber noch mehr hervor, die ermittelten Zeiten liegen folglich deutlich über den optimalen Werten. Da in Kapitel 3.4 bereits eine gute und beweisbare Schranke für diese Art von Problemen vorgestellt wurde, wurde dieser Ansatz nicht weiter verfolgt.

4.6 Heuristische Bestimmung des kombinatorischen Anteils mit physikalischen Splines

Physikalische Splines können aufgrund ihrer Ähnlichkeit zu einem MTSP auch gut zur heuristischen Bestimmung des kombinatorischen Anteils eines solchen verwendet werden. Dazu wird wieder der Branch & Bound Algorithmus von Dakin eingesetzt, zu minimieren ist nun allerdings die Endzeit t_f eines physikalischen Splines.

Städte	Durchschnitt	Median	Varianz
4	35,42	23,19	733,91
5	46,56	48,30	943,21
6	159,95	137,89	1139,20

Tabelle 4.5: Rechenzeiten in Sekunden für die Bestimmung einer Startlösung II

Zur Beurteilung der nötigen Rechenzeiten wurde analog zu Abschnitt 3.2 vorgegangen: Es wurden je 100 Probleme mit 4 bis 6 Städten zufällig erzeugt und anschließend gelöst. Auf die Lösung von Problemen mit 7 Städten wurde aufgrund der bedeutend höheren Rechenzeiten des Verfahrens verzichtet. Die verwendete CPU war auch in diesem Fall ein Intel Core 2 Duo mit 3 Gigahertz Taktfrequenz.

Tabelle 4.5 zeigt die durchschnittliche Rechenzeit, den Median (50% Quantil) sowie die Varianz in Sekunden [*sec*]. Wie man sieht, liegen alle Zeiten deutlich über denen des Verfahrens aus Kapitel 3.2. Dies ist darauf zurückzuführen, dass in jedem Knoten des B&B Baums ein Optimierungsproblem gelöst und nicht ein einfacher Index berechnet werden muss.

Kapitel 5

Fazit und Ausblick

In dieser Arbeit wurden Verfahren zur Bestimmung von Startschätzungen und Schranken für motorisierte Handlungsreisendenprobleme (MTSPe) entwickelt. Beide Größen sollten dabei möglichst exakt und möglichst effizient zu berechnen sein.

Zur Bestimmung einer Startschätzung für den kombinatorischen Anteil eines MTSP wurde ein Branch & Bound Verfahren auf Basis der Eigenschaften einer optimalen Trajektorie abgeleitet, dessen Rechenzeiten für praktisch lösbare MTSP Instanzen akzeptabel sind.

Die Bestimmung der minimalen Fahrzeit für eine gegebene Kurve ist ein nichttriviales Optimalsteuerungsproblem. Für Probleme ohne Geschwindigkeitsbeschränkung wurden zwei Approximationsverfahren für die minimale Fahrzeit entwickelt und deren Eignung als untere Schranken untersucht. Beide eignen sich gut zur Approximation der Fahrzeiten, aber allgemein nicht als untere Schranken. Für die in Kapitel 4 beschriebenen physikalischen Splines konnte eine qualitative Aussage getroffen werden, wann sie gut und wann weniger gut funktionieren.

Beide können auch gut im Rahmen eines Branch & Bound Verfahrens zur Bestimmung einer Startschätzung für den kombinatorischen Anteil eines MTSP eingesetzt werden. Für die physikalischen Splines wurden die dazu benötigten Rechenzeiten in Abschnitt 4.6 ermittelt.

Zudem wurden sowohl für Probleme mit als auch ohne Geschwindigkeitsbeschränkung beweisbare untere Schranken entwickelt. Allerdings ist nur die Schranke für Probleme mit Geschwindigkeitsbeschränkung ausreichend scharf, um sinnvoll in einem Branch & Bound Verfahren zur Lösung von MTSPen eingesetzt zu werden.

Literaturverzeichnis

[AMH02] Akenine-Möller, Tomas und Eric Haines: *Real-Time Rendering*. A.K. Peters Ltd, 2. Auflage, 2002.

[BGH+02] Buss, M., M. Glocker, M. Hardt, O. von Stryk, R. Bulirsch und G. Schmidt: *Nonlinear hybrid dynamical systems: modeling, optimal control, and applications*. In: S. Engell, G. Frehse, E. Schnieder (Herausgeber): *Modelling, Analysis and Design of Hybrid Systems*, Band 279 der Reihe *Lecture Notes in Control and Information Sciences (LNCIS)*, Seiten 311–335, Berlin, Heidelberg, 2002. Springer-Verlag.

[BP03] Bussieck, M. und A. Pruessner: *Mixed-Integer Nonlinear Programming*. Technischer Bericht, GAMS Development Corporation, 19. Februar 2003.

[BSM05] Bronstein, Ilja N., Konstantin A. Semendjajew und Gerhard Musiol: *Taschenbuch der Mathematik*. Verlag Harri Deutsch, 6. Auflage, 2005.

[Dak97] Dakin, R.: *Dakin Branch & Bound TSP Algorithm*. `http://www.pcug.org.au/dakin/tspbb.htm`, 1997.

[DB78] De Boor, Carl: *A practical guide to splines*. Springer, 1. Auflage, 1978.

[Dom95] Domschke, W.: *Logstik: Transport*. Oldenbourg, München, 4. Auflage, 1995.

[Dom97] Domschke, W.: *Logstik: Rundreisen und Touren*. Oldenbourg, München, 4. Auflage, 1997.

[Fou05] Fourer, Robert: *Nonlinear Programming - Frequently Asked Questions*. `http://www-unix.mcs.anl.gov/otc/Guide/faq/nonlinear-programming-faq.html`, 2005.

[Glo05] Glocker, Markus: *Diskret-kontinuierliche Optimalsteuerung: Modellierung, Numerik und Anwendung bei Mehrfahrzeugsystemen (Discrete-Continuous Optimal Control: Models, Numerical Methods and Applications in Multi-Vehicle Systems)*. Fortschritt-Berichte VDI, Technische Universität Darmstadt, 21. Oktober 2005.

[GMS02] Gill, Philip E., Walter Murray und Michael A. Saunders: *SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization.* SIAM Journal on Optimization, 12(4):979–1006, 2002.

[GvS02] Glocker, M. und O. von Stryk: *Hybrid optimal control of motorized traveling salesmen and beyond.* In: *Proc. 15th IFAC World Congress on Automatic Control*, Seiten 987–992, Barcelona, Spain, July 21-26 2002. Elsevier Science.

[Hei98] Heim, Alexander: *Modellierung, Simulation und optimale Bahnplanung bei Industrierobotern.* Doktorarbeit, Technische Universität München, 1998.

[HMS06] Holzmann, Günther, Heinz Meyer und Georg Schumpich: *Technische Mechanik Kinematik und Kinetik.* Teubner Verlag, 2006.

[KB84] Kochanek, Doris H. U. und Richard H. Bartels: *Interpolating Splines with Local Tension, Continuity, and Bias Control.* Computer Graphics (SIGGRAPH '84 Proceedings), Seiten 33–41, Juli 1984.

[Ley93] Leyffer, S.: *Deterministic Methods for Mixed Integer Nonlinear Programming.* Doktorarbeit, University of Dundee, 1993.

[LK73] Lin, S. und B. W. Kernighan: *An Effective Heuristic Algorithm for the Traveling-Salesman Problem.* Operations Research, 21(2):498–516, 1973.

[LMSK63] Little, J., K. Murty, D. Sweeney und C. Karel: *An Algorithm for the Traveling Salesman Problem.* Operations Research, 11(6):972–989, 1963.

[Mat02] MathWorks, The: *Cubic Spline Interpolation Demo.* `http://www.mathworks.com/products/plines/demos.html?file=/products/demos/shipping/splines/csapidem.html`, 2002.

[Sch01] Schöning, Uwe: *Theoretische Informatik - Kurzgefasst.* Spektrum Akademischer Verlag GmbH, 4. Auflage, 2001.

[Sch04] Schittkowski, Klaus: *Zur Stabilität von SQP-Verfahren und zur Lösung globaler und gemischt-ganzzahliger Optimierungsprobleme.* `http://www.dynardo.de/Presentations/StabilitaetSQP-Verfahren.pdf`, 2. Dezember 2004. Präsentation, 1. Weimarer Optimierungs- und Stochastiktage.

[vSG00] Stryk, O. von und M. Glocker: *Decomposition of mixed-integer optimal control problems using branch and bound and sparse direct collocation.* In: S. Engell, S. Kowalewski, J. Zaytoon (Herausgeber): *ADPM 2000 - The 4th International Conference on Automation of Mixed Processes: Hybrid Dynamic Systems*, Seiten 99–104, Aachen, 18. September 2000. Shaker.

[Wei02] Weisstein, Eric W.: *Cubic Spline*. `http://mathworld.wolfram.com/CubicSpline.html`, 2002.

[Wü97] Wünsch, Volkmar: *Differentialgeometrie - Kurven und Flächen*. Teubner Verlag, 1. Auflage, 1997.

Anhang A

Inhaltsverzeichnis der CD-ROM

Auf der beigefügten CD-ROM befinden sich die folgenden Verzeichnisse:

`/matlab`	MATLAB Quellcode der implementierten Verfahren
`/mtsp`	Daten der untersuchten MTSP Instanzen
`/quellen`	Alle im PDF-Format vorliegenden Quellen
`/tabellen`	Verwendete Excel 2003 Tabellen
`/tex`	LaTeXQuellcode dieser Diplomarbeit
`/zeichnungen`	Verwendete Visio 2003 Zeichnungen

Anhang B

Hinweise zur Implementierung

Für die Implementierung der in dieser Arbeit geschilderten Verfahren wurde MATLAB in der Version 7.2 (R2006a) für Windows verwendet. Zur Optimierung wurden Komponenten der MATLAB Optimization Toolbox in der Version 3.0.4 und die SNOPT Studentenversion (`http://cam.ucsd.edu/~peg/software.html`) verwendet.

Alle relevanten Quellcodes finden sich im Verzeichnis `/matlab` auf der beigefügten CD-ROM. In diesem Anhang werden die Quellcode-Dateien den entsprechenden Kapiteln zugeordnet.

Kapitel 3.1.2

`CubicHermiteInterpolation2D.m`
⇒ Implementierung der kubischen Hermite Interpolation
`CubicHermiteInterpolation2Dder1.m`
`CubicHermiteInterpolation2Dder2.m`
⇒ Erste und zweite Ableitungen der kubischen Hermite Interpolation
`CardinalSpline.m`
⇒ Implementierung eines kardinalen Splines
`KBSpline.m`
⇒ Implementierung eines Kochanek-Bartels Splines

Kapitel 3.2

`BBSolverOmega.m`
⇒ Implementierung des vollständigen Verfahrens
`BBBenchmarkOmega.m`
⇒ Benchmarkfunktion

Kapitel 3.3.2

`ApproximateDrivingTime.m`
⇒ Implementierung des Approximationsverfahrens

Kapitel 3.4

`lowerBoundVlimit.m`
⇒ Implementierung der verbesserten Schranke

Kapitel 3.5

`lowerBoundNoVlimit.m`
⇒ Implementierung der unteren Schranke

Kapitel 4.3

`PhysSplineFMINCON.m`
⇒ Bestimmung der Spline-Parameter mit `fmincon`
`PhysSplineSNOPT.m`
⇒ Bestimmung der Spline-Parameter mit `SNOPT`
`snoptuserfunphys.m`
⇒ Hilfsfunktion für die `SNOPT` Implementierung

Kapitel 4.6

`BBSolverPhysSpline.m`
⇒ Implementierung des Verfahrens
`BBBenchmarkPhysSpline.m`
⇒ Benchmarkfunktion