DHBW
Duale Hochschule
Baden-Württemberg
Mannheim

Duale Hochschule Baden-Württemberg Mannheim

Integrationsseminar

Traveling Salesperson Problem

Studiengang Wirtschaftsinformatik

Studienrichtung Data Science

Dozent:	Prof. Dr. Maximilian Scherer
Verfasser(in):	Ahmet Korkmaz, Henrik Rathai, Tobias Ludwig
Matrikelnummer:	6357368, 7843611, 7286643
Kurs:	WWI21DSA
Studiengangsleiter:	Prof. Dr.-Ing. habil. Dennis Pfisterer
Abgabedatum:	26.02.2024

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

ATSP	Asymmetrisches Traveling Salesperson Problem
MTSP	Multiples Traveling Salesperson Problem
NP	Nichtdeterministisch Polynomiell
STSP	Symmetrisches Traveling Salesperson Problem
TSP	Traveling Salesperson Problem
KI	Künstliche Intelligenz
ETH	Eidgenössische Technische Hochschule Zürich
KIT	Karlsruher Institut für Technologie

1 Einleitung

In der Ära der Automatisierung und fortschreitenden Entwicklung im industriellen Sektor spielt die Optimierung von Logistik und Lieferketten eine immer größere Rolle. Dies bietet verschiedene Anwendungsbereiche für das Traveling Salesperson Problem (TSP). Dieses mathematische Problem ist ein intensiv erforschtes Problem der kombinatorischen Optimierung, welches eine fundamentale Herausforderung in der theoretischen Informatik und angewandten Mathematik darstellt. Dabei geht es darum, den kürzesten möglichen Weg, von einem Startpunkt über verschiedene Ziele bis zurück zum Startpunkt zu ermitteln.

Diese Arbeit zielt darauf ab, ein umfassendes Verständnis des TSP zu vermitteln, indem sie sich auf vier wesentliche Aspekte konzentriert: die Grundlagen der Visualisierung von geometrischen Problemen, die mathematischen Grundlagen des TSP einschließlich seiner NP-Schwierigkeit, die historische Entwicklung und Entstehung des Problems, sowie verschiedene Lösungsansätze.

Zunächst werden die Grundlagen der Visualisierung von geometrischen Problemen erörtert, um ein grundlegendes Verständnis in diesem Fachbereich zu schaffen. Angesichts der Tatsache, dass das Problem des Handlungsreisenden (TSP) eine geometrische Herausforderung darstellt, die die Ermittlung der kürzesten Route durch eine bestimmte Anzahl von Punkten zum Ziel hat, ist die Visualisierung ein entscheidendes Werkzeug für das Verständnis und die Analyse des Problems. Dafür werden verschiedene Grundlagen, wie die Perzeption, aber auch moderne Anwendungen wie in Computergrafiken und Animationen beleuchtet.

Danach wird die Entstehung und die Entwicklung des TSP und ihre Wichtigkeit für die heutige Welt untersucht. Hierbei wird auf verschiedene Meilensteine in der Entwicklung, die moderne Formulierung, die mathematische Entwicklung und auf verschiedene Einsatzgebiete eingegangen.

Anschließend wird ein tieferer Einblick in die mathematischen Grundlagen des TSP gegeben. Dieser Einblick umfasst die formale Definition des Problems, seine Einordnung in die Komplexitätstheorie und die Diskussion seiner NP-Vollständigkeit. Mit Hilfe dieser Analyse kann das TSP im Kontext anderer komplexer Probleme der Informatik und Mathematik

eingeordnet werden, um die Herausforderungen bei der Suche nach effizienten Lösungen zu verstehen.

Schließlich werden unterschiedliche Strategien zur Lösung des TSP präsentiert. Diese reichen von exakten Methoden, über heuristische und metaheuristische Ansätze, bis hin zu modernen Methoden, die auf maschinellem Lernen und künstlicher Intelligenz basieren. Dies veranschaulicht die aktuelle Relevanz und Komplexität des Problems und gibt einen Einblick in aktuelle Lösungsansätze.

Schlussendlich zielt diese Arbeit darauf ab, das TSP von seinen Anfängen und mathematischen Prinzipien bis hin zu modernen Lösungsstrategien umfassend zu beleuchten und zu analysieren.

2 Visualisierung

In diesem Abschnitt werden die Grundprinzipien der Perzeption sowie die Anwendung von Visualisierungen im Rahmen der Problemlösung thematisiert. Das Ziel hierbei ist die Wahrnehmung von Visualisierungen zu verstehen und deren Einsatz zu erläutern.

2.1 Perzeption

Perzeption wird üblicherweise als die Wahrnehmung definiert, wobei sich die Bedeutung je nach Kontext unterscheiden kann. Im Kontext der Visualisierung bezieht sich die Perzeption im Wesentlichen auf die kognitive Verarbeitung von sensorisch vermittelten Informationen, und somit um das Sehen an sich. Dabei geht es nicht um das reine Erfassen von sensorischen Reizen, sondern auch um das Erkennen beziehungsweise Wiedererkennen von Wahrgenommenen Objekten oder Phänomenen. So werden zum Beispiel aus erkannten Punkten und Linien Formen und Objekte erkannt [1]. Hierbei beginnt die Perzeption im menschlichen Auge, wobei Lichtwellen, die von Objekten reflektiert werden, durch die Hornhaut, Pupille und Linse fallen und schließlich auf die Netzhaut treffen [2]. Die Netzhaut des menschlichen Auges enthält Fotorezeptoren, welche aus Zapfen und Stäbchen bestehen, die die ankommenden Lichtwellen in elektrische Signale umwandeln. Diese Signale werden über den Sehnerv an das Gehirn weitergeleitet, welches die Informationen verarbeitet und interpretiert [3]. Im Gehirn kommen verschiedene Regionen zum Einsatz, vor allem die Okzipitallappen. Diese übernehmen den Teil der Auswertung der Informationen und arbeiten hierbei mit verschiedenen anderen Bereichen zusammen, um eine vollständige visuelle Wahrnehmung zu ermöglichen. Dabei wird nicht nur das gesehene Objekt identifiziert, sondern auch auf seine Eigenschaften wie Licht, Kontrast, Form, Größe, Position, Farbe und Bewegung untersucht [4].

Bei der räumlichen Wahrnehmung benutzt das Gehirn verschiedene Methoden. Durch stereoskopisches Sehen, bei dem die Augen des Menschen nicht exakt dasselbe sehen, ist es dem Gehirn möglich, eine dreidimensionale Darstellung der Welt zu generieren. Weitere Aspekte wie der Texturgradient, Bewegungen der Objekte und die Geschwindigkeit ihrer Bewegungen auf der Netzhaut tragen zur räumlichen Wahrnehmung bei [4].

2.2 Moderne Anwendung

Die Visualisierung in der Computergrafik und Animation spielt eine entscheidende Rolle beim Verständnis wissenschaftlicher Probleme. Im Bereich der Computergrafik an Institutionen wie der ETH Zürich und dem KIT wird intensiv an der Analyse, Bearbeitung und Darstellung visueller Informationen gearbeitet [5]. Hierbei wird intensiv an der Erzeugung synthetischer Bilder und der interaktiven Manipulation für das Erstellen von Computeranimationen gearbeitet. Dadurch werden komplexe Daten anschaulich visualisiert, um diese dem Menschen verständlich zu machen. Dieses Vorgehen zielt darauf ab, eine optimale Visualisierung zu gewährleisten, die Analyse zu verbessern und die wesentlichen Informationen herauszufiltern. [6].

Visualisierung spielt besonders bei dem Verständnis komplexer Probleme eine große Rolle. Mit ihrer Hilfe können Informationen und komplexe Zusammenhänge in einer leicht erfassbaren Form dargestellt werden. Hierbei können Diagramme, Fotos, Zeichnungen, Schaubilder oder Infografiken präsentiert werden, um dem Menschen Zusammenhänge verständlicher zu machen. Effektive Visualisierungen lenken die Aufmerksamkeit auf das Wesentliche und heben wichtige Informationen hervor, beispielsweise durch den Einsatz von Farben. Sie vereinfachen die Gruppierung und Differenzierung von Informationen und stellen klare Zusammenhänge her. Wiederkehrende Elemente in Visualisierungen schaffen einen roten Faden, der die Aufnahme und das Verständnis von Informationen unterstützt [7].

3 Mathematischen Grundlagen

Mathematisch wird das TSP meist als ein klassisches Problem der Graphentheorie formuliert. Dabei besteht ein Graph aus Knoten (Städte) und Kanten (Wege zwischen den Städten), wobei jeder Kante ein Gewicht wie die Entfernung oder Kosten des Weges zugeordnet wird.

Um den Abstand zwischen zwei Punkten in einem Koordinatensystem zu berechnen, wird üblicherweise die euklidischen Distanzformel verwendet [8]. Diese besagt, wenn die Punkte $P_1(x_1, y_1)$ und $P_2(x_2, y_2)$ in einem zweidimensionalen Raum liegen, lautet die Formel für die Distanz d zwischen P_1 und P_2 [9]:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

In einem dreidimensionalen Raum mit den Punkten $P_1(x_1, y_1, z_1)$ und $P_2(x_2, y_2, z_2)$ wird die Formel wie folgt dargestellt:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}.$$

Das Ziel des TSPs ist es, in dem vorgestellten Graphen, einen Hamiltonschen Zyklus mit der minimalen Gesamtlänge zu finden. Ein Hamiltonscher Zyklus ist ein Pfad, der jeden Knoten genau einmal besucht und zum Startknoten zurückkehrt. Um den kürzesten Weg zu finden, kann beispielsweise die Brute-Force-Methode verwendet werden. Dabei wird jeder mögliche Weg zwischen den Punkten berechnet, die Gesamtlänge verglichen und so der kürzeste Weg ermittelt [10]. Als Vorstufe des Hamiltonschen Zyklus wird die eulerische Runde bezeichnet. Diese Runde bezeichnet einen Pfad, der jede Kante genau einmal durchläuft und zum Startpunkt zurückkehrt. Ein Graph, der eine eulersche Runde ermöglicht, muss zusammenhängend sein, und alle Knoten müssen einen geraden Grad haben. Die Brute-Force-Methode ist allerdings nur für Graphen mit wenig Punkten sinnvoll, da die Anzahl der möglichen Routen faktoriell mit der Anzahl der Städte ansteigt, was einen nicht tragbaren hohen Aufwand in Form von Kosten und Zeit zur Folge hätte.

Daher wird das TSP als NP-vollständig eingestuft, da es zu den Problemen gehört, für die bisher kein Algorithmus bekannt ist, der eine Lösung in polynomialer Zeit finden kann,

wenn die Größe des Problems über eine bestimmte Grenze hinausgeht. Die Klassifizierung als NP-vollständig weist darauf hin, dass das Problem nicht nur in der Kategorie NP liegt – was bedeutet, dass eine vorgelegte Lösung des Problems in polynomialer Zeit verifiziert werden kann –, sondern auch, dass es mindestens so schwer zu lösen ist wie das schwerste Problem in NP. Sollte es gelingen, einen polynomialen Algorithmus für ein NP-vollständiges Problem zu finden, würde dies implizieren, dass $P = NP$ ist. Dies würde bedeuten, dass alle in NP liegenden Probleme effizient lösbar wären. Hierbei handelt es sich jedoch um ein zentrales offenes Problem in der theoretischen Informatik und die weit verbreitete Annahme ist, dass $P \neq NP$, was bedeutet, dass es sehr unwahrscheinlich ist, dass effiziente Algorithmen für NP-vollständige Probleme existieren [11].

Diese Einordnung unterstreicht die Komplexität des TSP und erklärt, warum es als eines der herausforderndsten Probleme in der Informatik gilt. Die NP-Vollständigkeit signalisiert nicht nur die theoretische Schwierigkeit, eine effiziente Lösung zu finden, sondern betont auch praktische Limitationen: Bei zunehmender Anzahl von Knoten im Graphen steigt der Rechenaufwand zur Problemlösung exponentiell an, was bei größeren Instanzen des Problems zu einem nicht praktikablen Aufwand in Bezug auf Zeit und Ressourcen führt.

Das bedeutet, es ist unwahrscheinlich, dass es einen effizienten Algorithmus gibt, der das Problem in polynomieller Zeit lösen kann.

Aufgrund der Schwierigkeit, das TSP exakt zu lösen, werden oft heuristische oder approximative Methoden eingesetzt. Diese Methoden können in einer schnelleren Zeit gute, aber nicht unbedingt optimale Lösungen finden. Beispiele sind der Nearest-Neighbor-Algorithmus, Christofides-Algorithmus, Branch-and-Bound oder andere genetische Algorithmen. Diese werden in Kapitel fünf weiter ausgeführt [11].

Des Weiteren kann das TSP in drei verschiedenen Varianten auftreten.

Bei dem symmetrischen Traveling Salesperson Problem (STSP) ist die Entfernung zwischen den beiden Knoten in beide Richtungen gleich lang, was einen ungerichteten Graphen zur Folge hat. Durch die Symmetrie wird das Problem vereinfacht, da sich die Anzahl der zu berücksichtigen Routen reduziert. Das Ziel bleibt die die kürzeste Hamilton Route zu finden.

Hingegen bei dem asymmetrischen Traveling Salesperson Problem (ATSP) ist die Entfernung zwischen zwei Knoten unterschiedlich. Dies ist in der Realität deutlich öfter der Fall,

da Straßen nicht immer parallel zueinander verlaufen. Hier müssen Gewichtungen für beide Richtungen der Knoten separat berechnet werden, was einen höheren Rechenaufwand zur Folge hat. Zudem wird die Komplexität erhöht, da es auch die Möglichkeit gibt, dass es nur einen Pfad in eine Richtung gibt.

Eine Erweiterung des klassischen TSP ist das multiple Traveling Salesperson Problem (MTSP). Beim MTSP werden statt einer Person mehrere Personen betrachtet. Jeder Reisende startet von einem zentralen Standort (oft als Depot bezeichnet) und kehrt dorthin zurück. Das Ziel ist es, eine Reihe von Routen für die Verkaufspersonen zu planen, sodass jede Stadt genau einmal von einem der Reisenden besucht und die Gesamtreiselänge minimiert wird. Diese Variante ist komplexer, da sie die Koordination mehrerer Routen erfordert [11], [12].

4 Geschichte

Dieses Kapitel befasst sich mit der Entstehung und dem Verlauf des TSP und geht dabei auf seine historischen Wurzeln, die moderne Formulierung und die NP-Vollständigkeit ein.

Das TSP hat eine lange und facettenreiche Geschichte in der angewandten Mathematik und Informatik. Seine Ursprünge lassen sich bis ins 19. Jahrhundert zurückverfolgen. Der irische Mathematiker W.R. Hamilton und der britische Mathematiker Thomas Kirkman formulierten das Problem mathematisch, wobei es zuerst in einem Handbuch für Reisende aus dem Jahr 1832 ohne mathematische Betrachtung erwähnt wurde [11].

Karl Menger, ein Mathematiker und Ökonom, brachte das TSP in den 1920er Jahren unter seinen Kollegen in Wien ins Gespräch und veröffentlichte 1932 “Das Botenproblem’". In den 1940er Jahren wurde das TSP von Statistikern wie Mahalanobis, Jessen, Gosh und Marks im Zusammenhang mit landwirtschaftlichen Anwendungen untersucht. Merrill M. Flood machte das TSP in den 1950er Jahren bei seinen Kollegen an der RAND Corporation populär. In dieser Zeit erschienen auch erste Lösungen für das TSP, unter anderem von George B. Dantzig, Fulkerson und Johnson im Jahr 1954. Diese Lösungsansätze bildeten die Grundlage für die Entwicklung weiterer Lösungsmethoden [12]. Mit dieser Grundlage werden bis heute neue Lösungsansätze erforscht und weiterentwickelt. Auf diese wird in Kapitel fünf näher eingegangen.

In den 1970er Jahren wurde die NP-Vollständigkeit des Problems festgestellt. Durch diese Erkenntnis wurde ein starkes Interesse an diesem Problem geweckt, wodurch immer mehr heuristische und approximierte Lösungsansätze erarbeitet wurden, da eine exakte Lösung für große Instanzen des Problems unpraktikabel ist [12].

Eine moderne Formulierung lässt sich in zwei Versionen aufteilen: als Optimierungsproblem (die Funktion-Version) und als Entscheidungsproblem. Bei Optimierungsproblem soll ein vollständiger und gewichteter Graph, der einen Hamilton-Kreis [13] bildet, gefunden werden, der jede vorgegebene Stadt einmal besucht und wieder zum Startpunkt zurückkehrt und dies mit der kleinstmöglich zurückgelegten Distanz. Das Entscheidungsproblem hat dieselbe Voraussetzung, nur wird eine Entscheidung getroffen, ob es einen Graphen gibt, der ein vorgegebenes Gewicht nicht überschreitet [11]. Diese Entscheidungen sind

beispielsweise relevant, wenn ein neuer Kunde bei einem Lieferanten anfragt und der Lieferant entscheiden muss, ob er diesen Kunden beliefern kann. Dabei können die Gewichte die zurückzulegende Strecke sein und das vorgegebene Gewicht die maximale zurücklegbare Zeit des Lieferanten an einem Tag.

5 Lösungsverfahren

Beim Lösen des TSP gibt es verschiedene Ansätze, die genutzt werden können. Je nach Anwendungsfall und Größe des Problems eignen sich unterschiedliche Verfahren. Es sind exakte und heuristische Verfahren bekannt [14], [15]. In diesem Kapitel werden beide Arten thematisiert und Lösungsverfahren vorgestellt.

5.1 Exakte Verfahren

Exakte Verfahren liefern die optimale Lösung, und somit die kürzeste Route im TSP. Dabei ist die Laufzeit jedoch meist exponentiell [14], [15]. Im Folgenden werden vier exakte Lösungsansätze vorgestellt und erklärt.

5.1.1 Brute-Force

Brute-force ist ein exaktes Verfahren, das alle möglichen Lösungen eines Problems durchprobiert und die optimale Lösung findet. Dabei werden alle potenziellen Wege durchprobiert und die kürzeste Route ausgewählt. Vorteil dieser Methode ist, dass die optimale Lösung gefunden wird. Bei n vielen Knoten gibt es $(n-1)!$ mögliche Wege, weswegen die Methode sich nicht für große Probleme eignet. Die Laufzeit steigt dabei exponentiell mit der Anzahl der Knoten [16].

Fatimah, Syafiin und Fauzi [17] haben in 2021 drei verschiedene Verfahren zur Lösung des TSP verglichen. In ihrem Vergleich mit insgesamt vier verschiedenen Städten fanden sie heraus, dass die vollständige Aufzählung, neben dem Branch-and-Bound-Algorithmus, die optimale Lösung liefert und bevorzugten diese beiden Verfahren gegenüber dem Greedy-Algorithmus.

5.1.2 Branch-and-Bound-Algorithmus

Der Branch-and-Bound-Algorithmus ist eines der ältesten und am häufig verwendeten Verfahren zur Lösung des TSP [16]. Das Problem wird dabei in Teilprobleme aufgeteilt, deren zulässige Lösungen die gesamten zulässigen Lösungen des ursprünglichen Problems umfassen [18]. Diese Teilmengen stellen die „Branches“ dar [16]. Der Algorithmus wählt in jeder Iteration ein Teilproblem aus und löst dieses. Dabei wird die untere Schranke für den Wert der optimalen Lösung abgeleitet, falls diese mindestens so hoch ist, wie der Wert der bisher besten gefundenen realisierbaren Lösung. Alternativ kann das Problem auch die optimale Lösung enthalten, oder keine realisierbare Lösung vorzeigen. Falls das Teilproblem nicht eines der drei Ergebnisse liefert, wird es in neue Teilprobleme aufgeteilt. Die neuen Teilprobleme werden im Anschluss mit der gleichen Herangehensweise gelöst. Der Iterationsprozess wird so lange fortgesetzt, bis die Liste mit Teilproblemen leer ist [18].

Beim TSP verzweigt sich der Algorithmus somit in die Menge aller möglichen Wege, während die untere Schranke für die Länge des Weges für jede Teilmenge berechnet wird. Schließlich findet er eine einzelne Tour in einer Teilmenge, dessen Länge kleiner oder gleich einer unteren Grenze für jede Tour ist. Der Lösungsweg wächst dabei exponentiell in der Zeit, abhängig von der Anzahl der Knoten beziehungsweise Städte [14].

5.1.3 Branch-and-Cut-Algorithmus

Beim Branch-and-Cut wird die Grundstruktur des Branch-and-Bound beibehalten, indem eine Liste von Teilproblemen des ursprünglichen Problems verwaltet wird. Jedes dieser Teilprobleme repräsentiert einen potenziellen Lösungsweg und wird in einem Baum, dem sogenannten Branch-and-Cut-Baum, strukturiert. Die Knoten des Baumes werden dabei in drei Typen unterteilt: der aktuell bearbeitete Knoten, die noch nicht abgearbeiteten Knoten und die vollständig bearbeiteten Knoten [18].

Der Algorithmus durchläuft diesen Baum, indem er die Teilprobleme analysiert, lokale untere und globale obere Schranken berechnet und entsprechend verzweigt oder Teilprobleme verwirft (fathoming). Im Gegensatz zum Branch-and-Bound-Algorithmus, werden bei Branch-and-Cut zusätzlich Schnittebenen verwendet. Diese Schnittebenen sind durch global gültige Ungleichungen charakterisiert, die in jedem Knoten des Baumes angewendet werden. Sie dienen dazu, den Lösungsraum gezielt einzuschränken und somit effizienter

zu gestalten. Dadurch ermöglicht Branch-and-Cut eine tiefere und spezifischere Steuerung des Lösungsprozesses durch die Einbeziehung dieser zusätzlichen mathematischen Bedingungen [18]. Jedoch wird dieser Algorithmus auch ineffizient, wenn die Anzahl der Städte steigt [14].

5.1.4 Concorde

Die Concorde-Methode wird als das schnellste exakte Verfahren zur Lösung des TSP gesehen. Das Verfahren wurde von Applegate, Bixby, Chvatal und Cook eingeführt und stellte den Rekord für das schnellste Lösen eines TSPs mit 15.112 Knoten auf [14]. Obwohl der Algorithmus im Jahr 2006 eingeführt wurde, wird das Verfahren heutzutage immer noch als der beste exakte Algorithmus zur Lösung des TSP angesehen [19], [20].

Das Verfahren basiert auf der Kombination von Branch-and-Cut-Methoden und Cutting-Planes-Algorithmen, um iterativ Teile der Suche zu entfernen, welche nicht Teil der optimalen Lösung sind [20]. Der Nachteil des Verfahrens ist, dass es nur für symmetrische TSPs geeignet ist [14].

5.2 Heuristische Verfahren

Heuristische Verfahren finden eine approximative Lösung, jedoch ist die optimale Lösung nicht garantiert. Die Laufzeit bei heuristischen Lösungsansätzen ist dabei kürzer als bei exakten Verfahren [14], [15]. In diesem Unterkapitel werden heuristische Algorithmen und Ansätze vorgestellt und näher erläutert.

5.2.1 Nearest-Neighbor-Algorithmus

Der Nearest-Neighbor-Algorithmus ist ein einfacher Greedy-Algorithmus, welcher sich zu Beginn eine zufällige Stadt auswählt und von dieser ausgehend den nächsten Nachbarn besucht. Der Algorithmus wählt dabei immer den nächsten Nachbarn aus, der noch nicht besucht wurde. Wenn keine unbesuchten Städte vorhanden sind, endet der Algorithmus [14]. Der nachfolgende Pseudocode zeigt den Algorithmus in seiner einfachsten Form [18].

Algorithmus 1 Nearest-Neighbor-Algorithmus

Require: $n \geq 0$ ▷ Die Anzahl der Knoten im Graphen
Ensure: Hamiltonscher Zyklus durch alle Knoten
  Wähle einen beliebigen Knoten j aus, setze $l = j$
  Erstelle die Menge $W = \{1, 2, \ldots, n\} \setminus \{j\}$
  while $W \neq \emptyset$ **do**
    Wähle einen Knoten $j \in W$ so, dass $c_{lj} = \min\{c_{li} | i \in W\}$
    Verbinde l mit j
    $W \leftarrow W \setminus \{j\}$
    Aktualisiere $l = j$
  end while
  Verbinde l mit dem in zu Beginn gewählten Knoten, um einen Hamiltonschen Zyklus zu bilden.

Trotz der Einfachheit des Algorithmus, kann seine Laufzeit quadratisch werden, was ihn für ein Problem mit sehr vielen Knoten ungeeignet macht. Zudem variieren die Qualitäten der Lösungen, basierend auf dem Startknoten, während die optimale Route oft nicht gefunden wird. Je nach Startknoten, kann die Lösung bis zu 10 Prozentpunkte variieren [18].

Eine Modifikation des Algorithmus, bei der zunächst der zehn-nächste Nachbar-Subgraph berechnet wird, kann die Laufzeit erheblich reduzieren und die Qualität der Lösungen verbessern. Jünger, Reinelt und Rinaldi [18] haben in 1995 das gleiche Problem mit dem Standard-Nearest-Neighbor-Algorithmus und der modifizierten Version gelöst. Während die Standardversion 15,3 Sekunden brauchte, konnte die modifizierte Version das Problem in 0,3 Sekunden lösen.

5.2.2 Christofides-Algorithmus

Der Algorithmus von Christofides ist ein heuristisches Verfahren, welches das vorliegende Problem in einen minimalen Spannbaum umwandelt. Anschließend wird ein minimales Gewichts-Matching für die Knoten mit ungeradem Grad erstellt und mit dem Spannbaum kombiniert [19]. Zuletzt wird aus dem kombinierten Graphen ein Euler-Zyklus gebildet, wobei besuchte Knoten vermieden werden. Dadurch entsteht ein Hamiltonscher Zyklus, der eine Lösung für das TSP darstellt. Dabei wird ein Weg erstellt, welcher jede Stadt genau einmal besucht und zum Ausgangsknoten zurückkehrt [19], [21].

5.2.3 Ant-Colony-Optimization-Algorithmus

Der Ant-Colony-Optimization-Algorithmus ist eine metaheuristische Methode zur Lösung des TSP. Er basiert auf dem Verhalten echter Ameisen in der Natur [16].

Jalali, Afshar and Marino [22] erklären, dass Ameisen auf dem Weg zurück zu ihrem Nest Pheromonspuren legen, nachdem sie Nahrung gefunden haben. Andere Ameisen folgen diesen Spuren, insbesondere, wenn sie zu Nahrung führen und verstärken die Spur, wenn sie selbst Nahrung finden. Spuren mit einer höheren Pheromonkonzentration werden wahrscheinlicher gewählt, sodass die Pheromonspur als Orientierungshilfe fungiert, um von anderen Ameisen entdeckte Nahrungsquellen zu lokalisieren. Eine Ameisenkolonie kann diese Spuren nutzen, um den kürzesten Weg zwischen dem Nest und einer Nahrungsquelle zu finden. Je mehr Ameisen einem Pfad folgen, desto mehr erhöht sich die Pheromonkonzentration, welches den Weg attraktiver für weitere Ameisen macht [22].

Im Kontext des TSP erstellt der Algorithmus Werte für die Stärke der Pheromonspur T_{ij} für jede Kombination von zwei Punkten beziehungsweise Städten [16]. Die Ameisen werden dabei in zufällig ausgewählten Städten platziert [19]. Jede „Ameise“ im Algorithmus sucht nach einer gültigen Lösung, indem sie zu verschiedenen Orten (Städten) zieht, die sie noch nicht besucht hat. Nachdem alle Orte besucht wurden, kehrt die Ameise zum Startort zurück und erstellt damit einen Pfad T. Die Wahrscheinlichkeit, dass eine Ameise sich von einem Ort zum anderen bewegt, wird durch die Pheromonkonzentration und einen heuristischen Wert η_{ij} beeinflusst, welcher mit dem inversen Abstand zwischen den Städten zusammenhängt. Die Formel für die Wahrscheinlichkeit wird dabei wie folgt berechnet [16]:

$$P_{ij}^{k} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}$$

Der Algorithmus verwendet die Parameter α und β, um die relative Bedeutung der Pheromonspur im Vergleich zum heuristischen Wert zu steuern. Nachdem alle Ameisen ihre Pfade abgeschlossen haben, werden die Pheromonspuren aktualisiert, wobei auch die Pheromonverdunstung berücksichtigt wird, die durch einen Koeffizienten p dargestellt wird. Die Aktualisierung der Pheromonspur findet wie folgt statt [16]:

$$\tau_{ij}^{k} = (1-\rho)\tau_{ij}^{k} + \Delta\tau_{ij}^{k}$$

In 2010 wurde ein verbesserter Ant-Colony-Optimization-Algorithmus von Gan, Guo, Chang

und Yi [23] vorgestellt, welcher die Ameisen im Algorithmus in 2 Arten teilt. Die „gewöhnlichen Ameisen“ suchen, wie im regulären Algorithmus, nach einer Lösung, während die „Scout-Ameisen“ die Wahrscheinlichkeit von Mutationen auf der derzeit optimalen Lösung zu berechnen. Dies geschieht auf Basis eines Pfad-Bewertungsmodells, wodurch die Scout-Ameisen basierend auf der berechneten Mutationswahrscheinlichkeit in der Umgebung der optimalen Lösung suchen. Dadurch wird die Effizienz und Robustheit des Algorithmus verbessert. Die Effizienz wird dabei anhand von Experimenten bestätigt. Die Ergebnisse stellen bei einer Nutzung der Scout-Ameisen eine schnellere Konvergenz und schnellere Findung der optimalen Lösung dar, ohne in lokalen Lösungen festzustecken [23].

5.2.4 Greedy-Algorithmus

Der Greedy-Algorithmus ist ein heuristischer Ansatz zur Lösung des TSP, indem schrittweise ein Weg erstellt wird. Dabei wird wiederholt die kürzeste Kante ausgewählt und dem Weg hinzugefügt. Zu Beginn werden die Kanten bei einem TSP mit n Knoten nach ihrer Länge sortiert. Anschließend wird die kürzeste Kante ausgewählt, welche die folgenden drei Bedingungen nicht verletzt [19]:

1. Die Kante darf keinen Zyklus mit weniger als n Kanten erzeugen
2. Das Hinzufügen der Kante darf den Grad eines Knotens nicht auf mehr als zwei erhöhen
3. Die gleiche Kante darf nicht zweimal hinzugefügt werden

Nach dem Hinzufügen der Kante, wird überprüft, ob der Weg bereits n Kanten enthält. Falls dies nicht der Fall ist, wird Kantenauswahl-Schritt wiederholt. Wenn der Weg n Kanten enthält, ist die Lösung erreicht [19].

Goyal [14] schlägt zudem in 2010 zudem einen nicht-deterministischen Greedy-Algorithmus vor, welches das TSP in polynomieller Zeit löst. Sein Ansatz verbindet den Warshall-Algorithmus mit selektiver Kanteneliminierung aus dem Graphen. Durch den Warshall-Algorithmus wird der minimale Hamiltonsche Zyklus gefunden, während die selektive Kanteneliminierung die Anzahl der Kanten reduziert.

Der Pseudocode für den Warshalls Algorithmus sieht wie folgt aus [14]:

Algorithmus 2 Warshalls Algorithmus

```
procedure WARSHALL
    for i = 0 to max − 1 do
        for j = 0 to max − 1 do
            if path[i][j] == 1 then
                for k = 0 to max − 1 do
                    if path[j][k] == 1 then
                        path[i][k] ← 1
                    end if
                end for
            end if
        end for
    end for
end procedure
```

Während der Algorithmus die Kanten durchgeht, wird die größte und noch nicht besuchte Kante entfernt. Dabei ist relevant, dass nach dem Entfernen der Kante immer noch alle Knoten erreicht werden können. Der Algorithmus endet, wenn alle Kanten des Graphens durchgegangen wurden [14].

Dieser Algorithmus hat jedoch, ähnlich wie bei anderen Greedy-Algorithmen, den Nachteil, dass nicht immer die optimale Laufzeit gefunden wird. Da Greedy-Algorithmen auf Lösungen mit lokalen Minima basieren, ist der Algorithmus nicht in der Lage die optimale Lösung zu finden, wenn das Entfernen der kürzesten Kante, die Lösung verbessert [14].

5.2.5 Lin-Kernighan-Heuristik

Die Lin-Kernighan-Heuristik wird als einer der besten heuristischen Algorithmen zur Lösung des TSP angesehen. Das Verfahren ist in der Lage effektiv eine optimale oder fast optimale Lösung für STSPs zu finden [14].

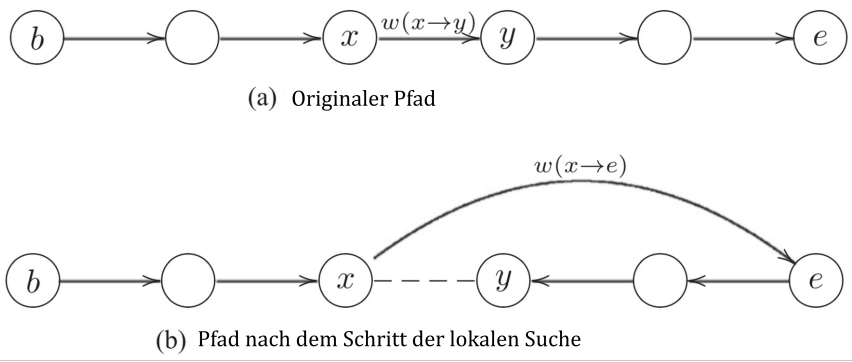
Die Lin-Kernighan-Heuristik löst das TSP, indem sie Kanten eines Weges entfernt und wieder hinzugefügt. Die Heuristik basiert auf der k-opt lokalen Suche, wodurch die Nachbarschaft der Wege untersucht werden. Durch das Entfernen und Hinzufügen von k Kanten werden nur die relevanten Teile der Nachbarschaft betrachtet. Nach der Entfernung einer Kante entsteht ein Weg, welcher umstrukturiert wird, um das Gewicht zu minimieren [24].

Der Ablauf als Pseudocode sieht dabei wie folgt aus [24]:

Algorithmus 3 Lin-Kernighan-Heuristik

```
Starte mit einer Tour T
repeat
    for jede Kante e in T do
        Erzeuge einen Pfad P durch das Entfernen der Kante e
        Strukturiere P um, damit das Gewicht minimiert wird
        if das Schließen von P zu einer Tour das Gewicht von T reduziert then
            Speichere die neue Tour als T
            Breche die Schleife ab und fange neu an
        end if
    end for
until keine Verbesserung mehr möglich ist
```

Durch dieses Verfahren entstehen neue Pfade. In Abbildung 5.1 ist beispielhaft ein Weg vor und nach der lokalen Suche dargestellt.



(a) Originaler Pfad

(b) Pfad nach dem Schritt der lokalen Suche

Abbildung 5.1: Pfade vor und nach der lokalen Suche
Quelle: Eigene Darstellung nach [24]

5.2.6 Genetische Algorithmen

Genetische Algorithmen werden zum Lösen von Optimierungsproblemen genutzt und orientieren sich an den Prinzipien der Evolution in der Natur. Die Algorithmen nutzen biologische Terminologie und arbeiten mit einer Population von Individuen, die durch Chromosomen repräsentiert werden. Chromosome, oder auch mathematische Gene, bestehen zum Beispiel aus einer Reihe von Zahlen in Binärform. Der Prozess eines genetischen Algorithmus besteht dabei aus drei Schritten [16]:

1. Erstellung einer Population
   Der Algorithmus erzeugt zufällig eine Reihe von Chromosomen, die die Anfangspopulation bilden. In der Regel besteht diese Population aus mindestens 50 Chromosomen.

2. Erstellung einer neuen Generation:
   In jeder neuen Generation haben Chromosomen mit einer geringeren Fitnessfunktion eine größere Überlebenschance. Durch Kreuzung und Mutation werden Chromosomen für die nächste Generation generiert.

3. Stop-Kriterium
   In jeder Generation wird der beste Wert der Fitnessfunktion gespeichert. Anschließend wird der 2. Schritt so lange wiederholt, bis keine Verbesserung im besten Fitnesswert nach aufeinanderfolgenden Generationen beobachtet wird und der Algorithmus somit endet.

Dieses Verfahren wird unter anderem bei der Lösung von MTSP genutzt. Dabei wird das Problem in mehrere einzelne TSPs aufgeteilt. Die einzelnen TSPs werden durch modifizierte genetische Algorithmen gelöst, sodass eine Lösung für das MTSP gefunden wird [15].

Bektas [15] zählt zudem weitere Lösungsansätze des TSPs durch genetische Algorithmen auf. Darunter auch der Ansatz von Song, K. Lee und W. Lee [25], welcher ein MTSP mit festen Kosten für jeden Verkäufer löst. In dem Ansatz wurde ein MTSP mit 400 Städten und drei Verkäufern verwendet, wobei der IBM PC-586 mit 400 MHz etwa 51 Minuten für die Lösung benötigte.

2021 wurde zudem ein Ansatz veröffentlicht, welcher das TSP mithilfe von künstlicher Intelligenz löst. Durch die Nutzung eines genetischen Algorithmus' und dem Hill-Climbing-Algorithmus, einem bekannten Optimierungsverfahren, konnten die Autoren ein industrielles Problem lösen. Das Problem wurde in Form eines TSPs formuliert, sodass die Lösung sich auf andere Probleme übertragen lässt. Die Autoren haben zunächst die Python Library mlrose genutzt, aber auch eigene Modifikationen an den Algorithmen vorgenommen, was zu Verbesserungen geführt hat. Der genetische Algorithmus wurde mit 100 Knoten und maximal 300 Generationen eingestellt. Die Ergebnisse zeigten, dass die modifizierte Version näher am Optimum war, jedoch aber auch mehr Laufzeit benötigte [26].

5.2.7 Künstliche Intelligenz

Künstliche Intelligenz wird bei der Lösung des TSPs seit mehreren Jahren genutzt. In 2015 erstellten Vinyals et al. [27] einen Ansatz, welcher ein sequenzielles Modell, basierend auf Long-Short-Term-Memory, nutzt. Der Algorithmus konstruiert iterativ eine Lösung, indem die Permutation der Eingabeknoten ausgegeben werden. Zudem wird der Concorde-Algorithmus für das Training genutzt. Neben diesem Ansatz listen Hudson et al. [20] insgesamt sieben Ansätze, welche das TSP lösen.

Hudson et al. [20] stellen auch ihren eigenen Ansatz dar, bei dem eine Kombination aus einem GNN (Graph Neural Network) und GLS (Guided Local Search) verwendet wird. Eine lokale Suche wird generell genutzt, um von einer Lösung aus iterativ benachbarte Lösungen durchzugehen, um eine optimalere Lösung zu finden. Basierend darauf, wird GLS als Metaheuristik genutzt, um lokale Optima eines Problems zu verlassen, indem bestimmte Aspekte der Lösung bestraft werden. Anschließend wird ein GNN genutzt, um den „Regret“ zu minimieren. Regret wird dabei als die Kosten der Einbeziehung der Entscheidung in die Lösung im Verhältnis zu einer optimalen Lösung gesehen. Das GNN bekommt die Länge der Kanten als Eingabe und gibt die Regret-Werte der Kanten aus. Ihre Ergebnisse zeigen eine zwei- bis sieben-fache Optimierung durch die Nutzung ihres Ansatzes.

6 Zusammenfassung

Im Rahmen dieser Arbeit wurden die Grundlagen der Visualisierung und deren Relevanz für das Verständnis von komplexen Problemen erklärt. Für eine Einführung in das TSP wurden dessen mathematischen Grundlagen, wie die Berechnung des kürzesten Weges zwischen Punkten, aber auch seine verschiedenen Arten betrachtet. Des Weiteren wurde die Entstehung des TSP und seine Entwicklung und Relevanz im Laufe der Jahre beleuchtet. Schlussendlich wurden verschiedene exakte und heuristische Lösungsansätze hervorgehoben und erklärt.

6.1 Fazit und Ausblick

Exakte Algorithmen bieten präzise Lösungen für das TSP, ihre Anwendung ist jedoch aufgrund exponentiell wachsender Rechenzeiten bei großen Problemstellungen limitiert. Im Gegensatz dazu ermöglichen heuristische und metaheuristische Verfahren praktikable Näherungslösungen für größere Instanzen, aber ohne Garantie auf die optimale Lösung. Zudem zeigt die Integration von KI-Technologien zukunftsträchtige Wege zur Effizienzsteigerung und Genauigkeit in der Lösungsfindung des TSP.

Zukünftige Forschungen könnten darauf abzielen, die Effizienz und Präzision bestehender Algorithmen weiter zu steigern, indem sie fortschrittliche Techniken der künstlichen Intelligenz integrieren. Die Herausforderung besteht darin, die Balance zwischen Berechnungseffizienz und Genauigkeit der Lösungen zu optimieren. Dabei ist es von entscheidender Bedeutung, innovative Methoden zu erforschen, die das Potenzial besitzen, die derzeitigen Begrenzungen bei der Bewältigung des TSP zu überwinden und neue Dimensionen der Problemlösung zu erschließen.

Literaturverzeichnis

[1] T. Krefeld. „Der perzeptiv-linguistische Forschungshorizont.“ (16. Apr. 2019), Adresse: https://www.dh-lehre.gwi.uni-muenchen.de/?p=135568&v=1 (besucht am 20. 12. 2023).

[2] M. F. Bear, B. W. Connors und M. A. Paradiso, „Das Auge,“ in *Neurowissenschaften: Ein grundlegendes Lehrbuch für Biologie, Medizin und Psychologie*, A. K. Engel, Hrsg. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, S. 309–347, ISBN: 978-3-662-57263-4. DOI: 10.1007/978-3-662-57263-4_9. Adresse: https://doi.org/10.1007/978-3-662-57263-4_9.

[3] H. A. Mallot, „Visuelle Wahrnehmung,“ *Handbuch der Allgemeinen Psychologie-Kognition*, S. 127–137, 2006.

[4] A. Leschnik, *Visuelle Wahrnehmung: Grundlagen, Clinical Reasoning und Intervention im Kindes- und Jugendalter* (essentials), 1. Aufl. Springer Wiesbaden, 2020, ISBN: 978-3-658-30876-6. DOI: 10.1007/978-3-658-30877-3.

[5] E. Zürich. „Visual Computing.“ (2023), Adresse: https://inf.ethz.ch/de/forschung/visual-computing.html (besucht am 20. 12. 2023).

[6] KIT. „Das IVD betreibt Forschung in den Bereichen Computergrafik, Visualisierung und Angewandte Geometrie.“ (2023), Adresse: https://www.ivd.kit.edu/ (besucht am 20. 12. 2023).

[7] C. Bingel. „Visualisierung fördert Lernen und Verstehen.“ (23. Apr. 2021), Adresse: https://www.haufe-akademie.de/blog/themen/coaches-und-berater/visualisierung-foerdert-lernen-und-verstehen/ (besucht am 20. 12. 2023).

[8] W. der BWL. „Euklidische Distanz.“ (), Adresse: https://welt-der-bwl.de/Euklidische-Distanz (besucht am 30. 12. 2023).

[9] P.-E. Danielsson, „Euclidean distance mapping,“ *Computer Graphics and image processing*, Jg. 14, Nr. 3, S. 227–248, 1980.

[10] S. Lin, „Computer Solutions of the Traveling Salesman Problem,“ *Bell System Technical Journal*, Jg. 44, Nr. 10, S. 2245–2269, 1965. DOI: https://doi.org/10.1002/j.1538-7305.1965.tb04146.x. Adresse: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1965.tb04146.x.

[11] J. Yu. „Traveling salesman problem.“ (2014), Adresse: https://optimization.cbe.cornell.edu/index.php?title=Traveling_salesman_problem (besucht am 20. 12. 2023).

[12] N. Cummings. „A brief History of the Travelling Salesman Problem.“ (Juni 2000), Adresse: https://www.theorsociety.com/about-or/or-methods/heuristics/a-brief-history-of-the-travelling-salesman-problem/ (besucht am 20. 12. 2023).

[13] K. Mönius, J. Steuding und P. Stumpf, „Wie man einen Städtetrip optimal plant,“ in *Algorithmen in der Graphentheorie : Ein konstruktiver Einstieg in die Diskrete Mathematik*. Wiesbaden: Springer Fachmedien Wiesbaden, 2021, S. 25–37, ISBN: 978-3-658-34176-3. DOI: 10.1007/978-3-658-34176-3_3. Adresse: https://doi.org/10.1007/978-3-658-34176-3_3.

[14] S. Goyal, „A Survey on Travelling Salesman Problem,“ 2010. Adresse: https://micsymposium.org/mics_2010_proceedings/mics2010_submission_51.pdf.

[15] T. Bektas, „The multiple traveling salesman problem: an overview of formulations and solution procedures,“ *Omega*, Jg. 34, Nr. 3, S. 209–219, 2006, ISSN: 0305-0483. DOI: https://doi.org/10.1016/j.omega.2004.10.004. Adresse: https://www.sciencedirect.com/science/article/pii/S0305048304001550.

[16] S. Sangwan, „Literature Review on Travelling Salesman Problem,“ *International Journal of Research*, Jg. 5, S. 1152, Juni 2018.

[17] I. A. S. Syafiin, S. N. Fatimah und M. Fauzi, „Travelling Salesman Problem Analysis with Complete Enumeration Method, Branch & Bound and Greedy Heuristic,“ *Eduvest - Journal of Universal Studies*, Jg. 1, Nr. 8, S. 752–756, 2021, ISSN: 2775-3735. DOI: 10.59188/eduvest.v1i8.144.

[18] M. Jünger, G. Reinelt und G. Rinaldi, „Chapter 4 The traveling salesman problem,“ in *Network Models*, Ser. Handbooks in Operations Research and Management Science, Bd. 7, Elsevier, 1995, S. 225–330. DOI: https://doi.org/10.1016/S0927-0507(05)80121-5. Adresse: https://www.sciencedirect.com/science/article/pii/S0927050705801215.

[19] R. Matai, S. Singh und M. Mittal, „Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches,“ in Nov. 2010, ISBN: 978-953-307-426-9. DOI: 10.5772/12909.

[20] B. Hudson, Q. Li, M. Malencia und A. Prorok, *Graph Neural Network Guided Local Search for the Traveling Salesperson Problem*. Adresse: http://arxiv.org/pdf/2110.05291.pdf.

[21] S. Nallaperuma, M. Wagner, F. Neumann, B. Bischl, O. Mersmann und H. Trautmann, „A Feature-Based Comparison of Local Search and the Christofides Algorithm for the Travelling Salesperson Problem," in *Proceedings of the Twelfth Workshop on Foundations of Genetic Algorithms XII*, Ser. FOGA XII '13, Adelaide, Australia: Association for Computing Machinery, 2013, S. 147–160, ISBN: 9781450319904. DOI: 10.1145/2460239.2460253. Adresse: https://doi.org/10.1145/2460239.2460253.

[22] M. R. Jalali, A. Afshar und M. MARINO, „Ant Colony Optimization Algorithm (ACO); A new heuristic approach for engineering optimization," Jg. 2, Mai 2005.

[23] R. Gan, Q. Guo, H. Chang und Y. Yi, „Improved ant colony optimization algorithm for the traveling salesman problems," *Journal of Systems Engineering and Electronics*, Jg. 21, Nr. 2, S. 329–333, 2010. DOI: 10.3969/j.issn.1004-4132.2010.02.025.

[24] D. Karapetyan und G. Gutin, „Lin–Kernighan heuristic adaptations for the generalized traveling salesman problem," *European Journal of Operational Research*, Jg. 208, Nr. 3, S. 221–232, 2011, ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2010.08.011. Adresse: https://www.sciencedirect.com/science/article/pii/S0377221710005485.

[25] C.-H. Song, K. Lee und W. Lee, „Extended Simulated Annealing for Augmented TSP and Multisalsemen TSP," Bd. 3, Aug. 2003, 2340–2343 vol.3, ISBN: 0-7803-7898-9. DOI: 10.1109/IJCNN.2003.1223777.

[26] S. Wintersteller, M. Uray, M. Lehenauer und S. Huber, *Improvements for mlrose Applied to the Traveling Salesperson Problem*, 2021. DOI: 10.1007/978-3-031-25312-6{\textunderscore}72. Adresse: http://arxiv.org/pdf/2109.14392.pdf.

[27] O. Vinyals, M. Fortunato und N. Jaitly, *Pointer Networks*, 2015. Adresse: http://arxiv.org/pdf/1506.03134.pdf.