Bachelorarbeit

Planung von Mittelspannungsnetzen mithilfe eines Ameisenalgorithmus

Jonathan Zeller

23. September 2021

Albert-Ludwigs-Universität Freiburg im Breisgau
Technische Fakultät
Institut für Informatik

Bearbeitungszeitraum
01. 07. 2021 – 01. 10. 2021

Gutachter
Prof. Dr. Hannah Bast

Betreuer
Wolfgang Biener
Janis Kähler
Matthias Hertel

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Breisach, 23.9.21
Ort, Datum

J. Zeller
Unterschrift

Inhaltsverzeichnis

Zusammenfassung

Diese Bachelorarbeit befasst sich mit der Planung von Mittelspannungsnetzen. Hierbei müssen die sogenannten Busse untereinander sowie mit einem Transformator verbunden werden. Busse sind elektrische Komponenten, an welche alle anderen relevanten Komponenten angeschlossen werden. Der Transformator liefert einen Großteil der Energie zur Versorgung der elektrischen Verbraucher. Diese Arbeit baut auf der Dissertation von Niklas Rotering auf. Sein Ansatz wird hierbei implementiert, erweitert und analysiert. Die Grundidee des Algorithmus besteht darin, einige Lösungen auf zufällige Art und Weise zu erzeugen. Die Lösungen werden anschließend bewertet und je nachdem wie gut die Lösungen sind erhalten die beteiligen Komponenten einen höheren Wert. Dieser Wert beeinflusst die Wahrscheinlichkeitsverteilung, nach welcher in der folgenden Iteration die Komponenten gewählt werden. Komponenten, welche an guten Lösungen beteiligt waren, werden dadurch häufiger gewählt. Mithilfe von dieser Heuristik soll nun in einem iterativen Verfahren eine möglichst gute Lösung gefunden werden.

1. Einleitung

Diese Bachelorarbeit startet mit einer Motivation, welche die Relevanz der Planung von Mittelspannungsnetzen zeigt. Anschließend gibt Abschnitt 1.2 den aktuellen Stand der Forschung wieder und Abschnitt 1.3 erläutert den Aufbau dieser Arbeit.

1.1. Motivation

Der Klimawandel ist eines der größten Probleme der heutigen Zeit. Um dieses Problem zu lösen, plant die Bundesregierung aktuell den Wechsel zur Elektromobilität sowie die vermehrte Nutzung von erneuerbaren Energien [1, 2, 3]. Beide Lösungswege haben eines gemeinsam. Sie sind dezentral und (besonders in Zukunft wohl) überall zu finden. Anstatt eines großen Kohlekraftwerks haben viele verschiedene Häuser eine Solaranlage auf dem Dach und zudem ein Elektroauto in der Garage. Dies führt dazu, dass sich die Anforderungen an das elektrische Stromnetz ändern. Anstatt den Strom von einem großen Erzeuger, wie bspw. einem Kohlekraftwerk, zu vielen Verbrauchern zu verteilen, gibt es jetzt sehr viele kleinere Einspeisungen und neue Verbraucher an allen möglichen Orten. Dadurch, dass die Autos von Verbrennungsmotoren auf einen elektrischen Antrieb wechseln, benötigt ein weiterer Stromabnehmer nun ebenfalls elektrische Energie. Zusätzlich zu den vielen dezentralen Einspeisungen kommt die Einspeisung von Windparks in der Nord- und Ostsee, welche von dort lange Strecken bis zum Verbraucher zurücklegen muss, teilweise bis in den Süden Deutschlands [2, 4, 5]. Für beide Szenarien sind die heutigen Stromnetze zum großen Teil nicht ausgelegt und müssen deshalb ausgebaut werden [4, 5]. Hierzu wurde bereits ein Verfahren vorgeschlagen [6], welches in dieser Arbeit erweitert, optimiert und analysiert wird.

1.2. Stand der Forschung

Diese Arbeit basiert auf der Promotion von Niklas Rotering, in welcher die Zielnetzplanung von Mittelspannungsnetzen behandelt wird [6]. In der Dissertation zeigt er ein Verfahren, mit welchem ein Mittelspannungsnetz möglichst kostengünstig gebaut werden kann. Hierzu werden die Koordinaten des Stromnetzes, der Verbraucher und der Versorger benötigt und daraus ein Graph aufgebaut. Auf diesem läuft dann ein heuristisches Verfahren, das sogenannte Ant Colony Optimization Verfahren,

ab. Dieses erzeugt in jeder Iteration ein Mittelspannungsnetz. Nach jeder Iteration wird ausgewertet, wie gut die Lösung der aktuellen Iteration ist und das Ergebnis in einer Datenstruktur vermerkt. In dieser Datenstruktur sammelt sich somit über die Iteration das Wissen, welche Komponenten häufig an guten Lösungen beteiligt sind. Dieses Wissen wird bei der Erzeugung von neuen Lösungen in den nachfolgenden Iterationen benutzt, wodurch der Algorithmus im Verlauf der Zeit meistens immer bessere Lösungen findet. Da es sich jedoch um ein heuristisches Verfahren handelt, kann dies nicht bzw. nur bedingt garantiert werden. Das Verfahren wurde in seiner Arbeit auf zwei Mittelspannungsnetze angewendet und hat hierbei gute Ergebnisse geliefert. Deshalb wird dieses Verfahren hier nun ebenfalls verwendet. In der folgenden Arbeit wird das oben beschriebene Verfahren implementiert, wobei neue Randfälle berücksichtigt werden. Das Verfahren wird erweitert und optimiert, um in mehr Situationen bessere Lösungen zu finden. Zudem wird das Verfahren sowohl praktisch als auch theoretisch analysiert.

Neben Ant Colony Optimization gibt es noch andere Verfahren zur Optimierung von Mittelspannungsnetzen, wie bspw. genetische Algorithmen oder stochastische Algorithmen [7]. Stochastische Algorithmen sind jedoch häufiger in der Planung mit mittelfristigen Zeithorizonten zu finden, bei welcher es darum geht, von dem aktuellen Stromnetz in das neue optimierte Stromnetz zu wechseln, ohne dass es während dem Übergang, zu Versorgungsengpässen kommt.

In dieser Arbeit wird jedoch Ant Colony Optimization verwendet, da dieses Verfahren auch bei dynamischen Problemen gut funktioniert [8]. So wurde Ant Colony Optimization z.B. erfolgreich auf das Network Routing Problem angewendet [8], welches ein hochdynamisches Problem ist. Bei dem Network Routing Problem, am Anwendungsfall des Internets, geht es darum, wie man Informationen weiterleitet, sodass sie möglichst schnell von einem Startpunkt zu einem Zielpunkt kommen. Hierbei hat jeder Knoten eine sogenannte Routingtabelle, worin steht, in welche Richtung der Knoten die Pakete weiterleiten muss, um das Paket näher zum Ziel zu senden. Da sich die Auslastung im Internet ständig und meist unvorhersehbar ändert und auch häufig mal Knoten ausgeschaltet werden, aber andere dafür plötzlich hinzugefügt werden, ist dieses Problem hochdynamisch. Hierbei hat die Heuristik des Ant Colony Optimization Algorithmus jedoch sehr gute Ergebnisse erzielt [8]. Die Robustheit gegenüber dieser hochdynamischen Situation ist der Grund dafür, dass in dieser Arbeit Ant Colony Optimization implementiert, erweitert und analysiert wird. Denn so kann, aufbauend auf dieser Arbeit, in einer zukünftigen Version die Positionierung der Verbraucher, genauer gesagt der Ortsnetzstationen (siehe Abschnitt 2.1), variiert werden, wodurch sowohl das Niederspannungsnetz als auch das Mittelspannungsnetz gleichzeitig optimiert werden können.

1.3. Aufbau der Arbeit

Das Problem der Mittelspannungsnetzplanung besteht darin, Verbraucher, Erzeuger sowie elektrische Bauteile wie z.B. Transformatoren miteinander zu verbinden. Die Verbraucher, Erzeuger und elektrischen Bauteile können in einem Graphen als Knoten repräsentiert werden. Die Knoten können nun beliebig miteinander verbunden werden. Die Anzahl an möglichen Kanten K, um N Knoten zu verbinden, beträgt also

$$K = \frac{N \cdot (N-1)}{2},$$

da jeder Knoten mit $N-1$ anderen Knoten verbunden sein kann. Da hierbei jedoch jede Kante doppelt gezählt wird, muss noch durch 2 geteilt werden. Für N Knoten gibt es somit

$$2^K = 2^{\frac{N \cdot (N-1)}{2}}$$

verschiedene Möglichkeiten, einen Graphen zu erzeugen, da jede Kante entweder Teil des Graph ist oder nicht. Wie man sieht, wächst die Menge an Graphen exponentiell mit der Anzahl an Knoten. Deshalb ist es nicht praktikabel, alle Möglichkeiten auszuprobieren, um das globale Optimum zu finden. Zudem müssen noch Nebenbedingungen beachtet werden, damit die Versorgungssicherheit gewährleistet ist und es möglichst selten zu Stromausfällen kommt. Das von Niklas Rotering vorgeschlagene Verfahren [6] reduziert zuerst den Lösungsraum und benutzt anschließend ein heuristisches Verfahren namens Ant Colony Optimization. Dieses Verfahren wird in dieser Bachelorarbeit implementiert, erweitert und analysiert. Hierzu werden in Kapitel 2 zuerst einige Grundlagen der Elektrotechnik vorgestellt. In Kapitel 3 wird anschließend das heuristische Verfahren der Ameisen Algorithmen vorgestellt. In Kapitel 4 wird dann der Algorithmus sowie die Erweiterungen am Algorithmus vorgestellt. Dieser Algorithmus wird im darauffolgenden Kapitel, Kapitel 5, theoretisch analysiert. Kapitel 6 evaluiert die Leistung des Algorithmus mithilfe von Mittelspannungsnetzen mit bekanntem globalen Optimum. Zudem wird der Algorithmus mit einem anderen Algorithmus für die Mittelspannungsnetzplanung verglichen. Kapitel 7 schließt diese Arbeit mit einer Zusammenfassung und einem Ausblick ab.

2. Grundlagen der Elektrotechnik

Dieses Kapitel enthält alle nötigen Grundlagen der Elektrotechnik, die für den darauffolgenden Teil der Bachelorarbeit wichtig sind. Es beginnt in Abschnitt 2.1 mit den Komponenten, mit welchen ein elektrisches Stromnetz aufgebaut ist. In Abschnitt 2.2 werde ich erklären, wie ein typisches Mittelspannungsnetz aufgebaut ist und zuletzt wird in Abschnitt 2.3 auf mögliche Steuerungsoptionen eingegangen, welche dem Netzbetreiber bzw. der Bundesregierung zur Verfügung stehen.

2.1. Ausgewählte Komponenten der Stromnetze

In diesem Kapitel werden die Komponenten der Stromnetze genauer erklärt, welche für den darauffolgenden Teil der Bachelorarbeit relevant sind.

Verbraucher sind an das Stromnetz angeschlossen und verbrauchen elektrische Energie, welche das Stromnetz zur Verfügung stellt. Beispiele für Verbraucher sind Fabrikanlagen, Haushaltsgeräte und Wärmepumpen wie Klimaanlagen oder Heizungen.

Erzeuger generieren elektrische Energie und stellen diese dem Stromnetz zur Verfügung. Je nachdem, wie viel Energie erzeugt wird, wird der Erzeuger an das Höchst-, Hoch-, Mittel- oder Niederspannungsnetz angeschlossen. Photovoltaikanlagen sind meist an das Niederspannungsnetz angeschlossen, während Kernkraftwerke hingegen an das Höchstspannungsnetz angeschlossen sind.

Speicher können elektrische Energie aus dem Stromnetz aufnehmen und zwischenspeichern und die Energie zu einem späteren Zeitpunkt wieder in das Stromnetz einspeisen. Dies ist besonders im Zusammenspiel mit Wind- und Solarenergie sinnvoll, da beispielsweise bei der Solarenergie mittags sehr viel Strom produziert wird. Der Überschuss davon kann dann zwischengespeichert werden und zu einem späteren Zeitpunkt, z.B. nachts, verwendet werden.

Leitungen verbinden die Komponenten der Stromnetze miteinander und sorgen dafür, dass elektrischer Strom zwischen ihnen fließen kann. Hierbei wird jedoch ein Teil der elektrischen Energie in Wärmeenergie umgewandelt, welche die Leitungen erwärmt. Dieser Energieverlust ist bei größeren Spannungen geringer. Aus diesem Grund ist die Spannung bei Leitungen, welche über große Distanzen gehen, höher.

Transformatoren sind elektrische Bauteile, welche der Spannungsumwandlung dienen. So kann ein Transformator z.B. 100 kV Wechselspannung aus dem Hochspannungsnetz erhalten und diese in 20 kV Wechselspannung für das Mittelspannungsnetz umwandeln. Für die Umwandlung von diesen 20 kV Wechselspannung zu den an den Haushalten üblichen 400 V wird ebenfalls ein Transformator benötigt. Die meisten Transformatoren können zudem, in einem gewissen Rahmen, die Spannungsumwandlung beeinflussen. Wenn die Spannung im Hochspannungsnetz z.B. etwas höher als normal ist, kann der Transformator diese kleinere Abweichung beheben.

Schalter dienen dazu, elektrische Komponenten je nach Situation zu verbinden oder zu trennen. Wenn der Schalter geschlossen ist, so sind die beiden Bauteile, an welche der Schalter angeschlossen ist, miteinander verbunden und Strom kann fließen. Ist der Schalter jedoch geöffnet, so sind die beiden Bauteile nicht miteinander verbunden.

Busse [1] sind Knotenpunkte, an welche die verschiedenen elektrischen Bauteile angeschlossen werden. Ein Bus verbindet also Generatoren und Verbraucher mit dem Stromnetz, er kann aber auch einfach nur mehrere Leitungen aus unterschiedlichen Richtungen miteinander verbinden.

Ortsnetzstationen sind Transformatoren, welche die Spannung des Mittelspannungsnetzes in die geeignete Spannung für das Niederspannungsnetz transformieren. Von einer Ortsnetzstation ausgehend werden die Endverbraucher über das Niederspannungsnetz versorgt (sofern es sich nicht um Großabnehmer aus der Industrie handelt, welche bei großem Energiebedarf an das Mittelspannungsnetz angeschlossen sind). Über eine Ortsnetzstation kann jedoch auch vom Niederspannungsnetz elektrische Energie zurück in das Mittelspannungsnetz fließen, falls dort z.B. wegen der Erzeugung erneuerbarer Energien ein Überschuss entsteht.

Stromnetz ist die Bezeichnung für ein Netzwerk, durch welches elektrische Energie fließt und zum Großteil aus den oben bereits genannten Bauteilen besteht. Das Stromnetzwerk nimmt elektrische Energie durch Erzeuger auf, leitet diese mithilfe von Leitungen weiter, bis die Energie schließlich bei den Verbrauchern verbraucht wird. Abbildung 2.1 zeigt ein exemplarisches Stromnetz, mit Erzeugern, Verbrauchern und den verschiedenen Spannungsebenen, welche jeweils über Transformatoren miteinander verbunden sind. Für diese Arbeit ist insbesondere das Mittelspannungsnetz relevant. Wie in der Abbildung zu sehen ist, erhält es die Energie vor allem aus dem Hochspannungsnetz, aber auch durch erneuerbare Energien wie Windräder oder größere Solaranlagen und kleineren Kraftwerken. Die Energie wird dann weitergeleitet zu größeren industriellen Abnehmern und den Ortsnetzstationen. Diese wandeln die Spannung für das Niederspannungsnetz um, um es zu versorgen. In der Abbildung ist jedoch nicht zu sehen, dass Solaranlagen auf den Hausdächern ebenfalls Energie erzeugen und in das Netz einspeisen. Diese kann, falls das Nie-

[1] Im deutschen wird häufiger das Wort Knoten verwendet. Um jedoch Verwechslungen mit den Knoten der Graphenteorie zu vermeiden, werde ich in dieser Arbeit den Begriff Busse verwenden.

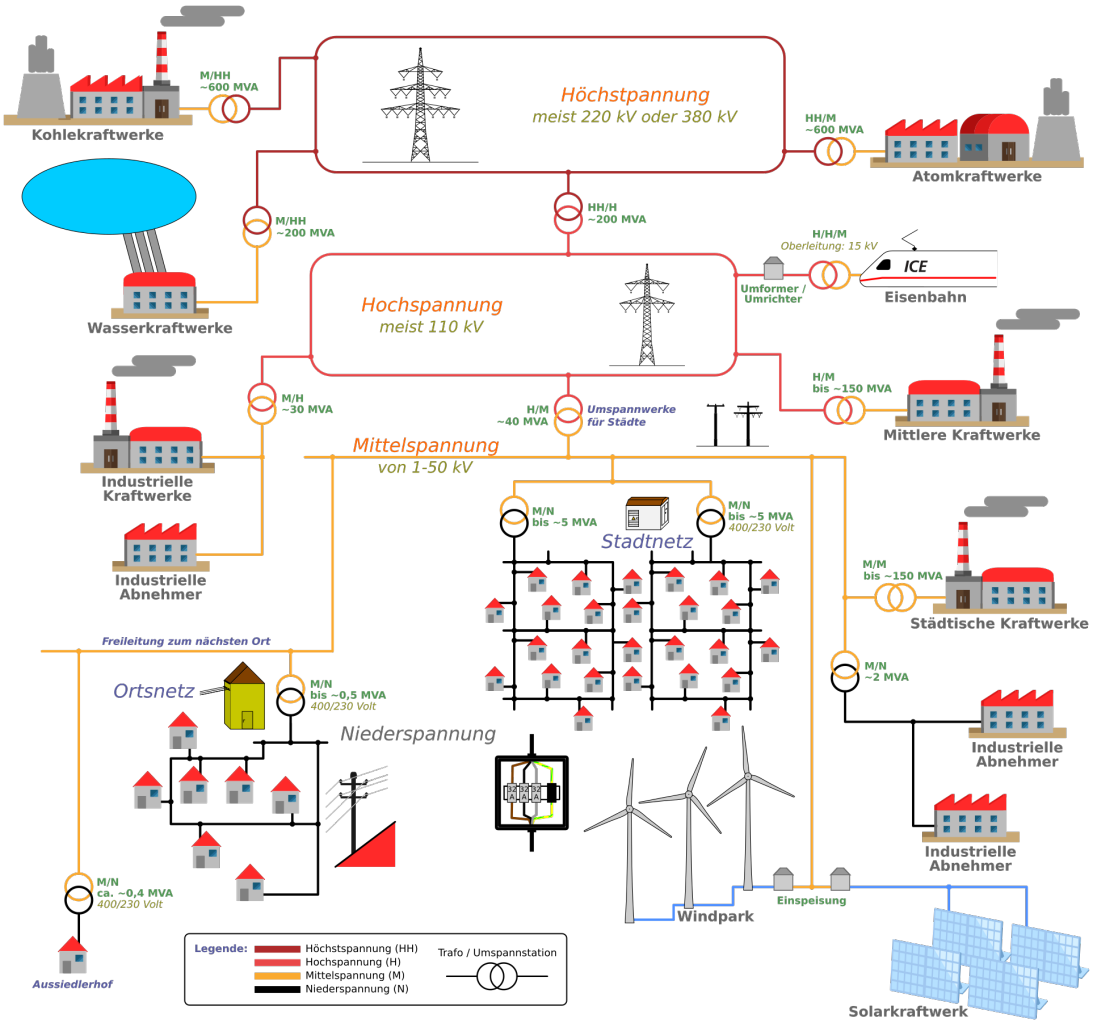


Abbildung 2.1.: Beispiel für ein Stromnetz mit verschiedenen Spannungsebenen. Quelle: Stefan Riepl, April 2018 [9]

derspannungsnetz keine weitere Energie benötigt, über die Ortsnetzstation zurück in das Mittelspannungsnetz geleitet werden. Dies passiert z.B. häufiger in Dörfern, welche keinen hohen Verbrauch haben, aber viel Strom über erneuerbare Energien produzieren.

2.2. Mittelspannungsnetze

Mittelspannungsnetze erhalten ihre elektrische Energie aus dem Hochspannungsnetz sowie von kleineren Kraftwerken oder teilweise auch aus Einspeisungen von erneuerbaren Energien (durch größere, direkt an das Mittelspannungsnetz angeschlossene Anlagen oder durch einen Überschuss im Niederspannungsnetz). Die Energie, welche

aus dem Hochspannungsnetz kommt, wird über einen Transformator in den gültigen Spannungsbereich für das Mittelspannungsnetz transformiert. Wie in Abschnitt 2.1 bereits erklärt, kann der Transformator hierbei kleine Spannungsänderungen beheben. Dadurch kann die komplette Struktur des Hochspannungsnetzes als ein Erzeuger abstrahiert werden und muss nicht gesondert betrachtet werden [6]. Für das Niederspannungsnetz gilt etwas Ähnliches. Dieses kann, je nachdem ob die elektrischen Komponenten mehr verbrauchen oder erzeugen, als Verbraucher oder als Erzeuger abstrahiert werden.

Die Struktur eines Mittelspannungsnetzes unterliegt gewissen Einschränkungen. Um die Versorgungssicherheit auch im Fehlerfall garantieren zu können gibt es das sogenannte N-1 Kriterium [6]. Dieses besagt, dass die Versorgung der Verbraucher auch dann noch erfüllt sein muss, wenn ein Bauteil ausfällt [6]. Aus diesem Grund sind in jeder Transformatorstation stets zwei Transformatoren [6]. Die Stromleitungen müssen hierbei so verlegt werden, dass ein Bus immer von zwei Seiten erreichbar ist, damit eine Leitung ausfallen kann. Dies ist genauer in Abbildung 2.2 zu sehen. Links oben sieht man die sogenannte Ringstruktur. Wenn Transformatoren und Busse in einem Graphen als Knoten repräsentiert werden, handelt es sich um einen Ring, wenn es einen Zyklus gibt, an dem ein Transformator beteiligt ist. Alle Komponenten, welche in dem Zyklus enthalten sind, gehören dann zum Ring. Im Normalbetrieb ist die Trennstelle geöffnet und der Weg vom Transformator zu jedem Bus somit eindeutig. Wenn jetzt jedoch beim Bus B_1 ein Fehler auftritt, so kann die Trennstelle geschlossen werden und die beiden Leitungen links und rechts vom fehlerhaften Bus B_1 werden vom restlichen Netz getrennt (siehe Schalter in Abschnitt 2.1). Auf diese Art und Weise können die beiden oberen Busse B_2 und B_3 weiterhin versorgt werden und nur der fehlerhafte Bus ist vom Netz getrennt. Falls eine Leitung gewartet oder ausgetauscht werden muss, ist es weiterhin möglich alle Busse mit Strom zu versorgen. Links unten ist die Struktur des Rings mit Stich zu sehen. Damit auch für die drei Stationen im Stich das N-1 Kriterium erfüllt ist, muss dort ein Notstromaggregat vorhanden sein [6]. Da dies natürlich nicht die optimale Lösung ist, ist die Länge eines Stichs auf höchstens drei Stationen beschränkt. Gerade in ländlichen Regionen ist es jedoch aus Kostengründen sinnvoll, wenn manche Busse per Stich an das Stromnetz angeschlossen sind. Dadurch können lange Strecken eher vermieden werden, was insbesondere wegen der Tiefbaukosten sinnvoll ist. In der Mitte der Abbildung ist die Struktur des Strangs zu sehen. Wenn Transformatoren und Busse als Knoten in einem Graphen repräsentiert werden, handelt es sich um einen Strang, wenn es einen Pfad von einem Transformator zu einem anderen Transformator gibt. Alle Knoten, welche an diesem Pfad beteiligt sind gehören dann zum Strang. Hier wird das N-1 Kriterium dadurch erfüllt, dass die elektrische Energie von einem der beiden Transformatoren kommen kann. Für das N-1 Kriterium reicht es jedoch nicht, jeden Bus über zwei Wege von Transformatoren erreichen zu können. Wenn z.B. der Bus B_1 ausfällt, wird die Trennstelle geschlossen und die Busse B_2 und B_3 werden ebenfalls über den Weg, welcher von unten kommt, versorgt. Dadurch fließt durch die Leitungen im unteren Teil des Netzes mehr Strom als üblich. Die Leitungen

müssen also in der Lage sein, diesen zusätzlichen Strom zu bewältigen.

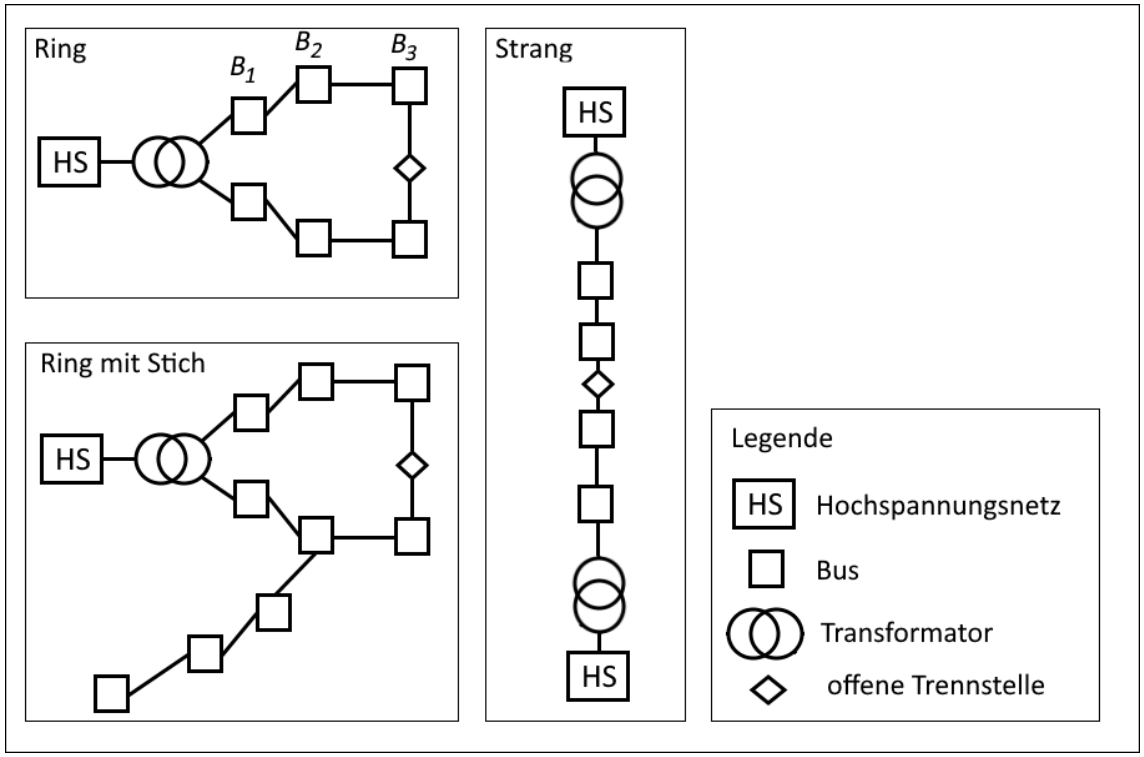


Abbildung 2.2.: Die erlaubten Netzstrukturen im Mittelspannungsnetz. Im Normalbetrieb geschlossene Trennstellen sind aus Gründen der Übersichtlichkeit nicht eingezeichnet. Quelle: Eigene Darstellung

2.3. Steuerungsoptionen

Durch Steuerungstechnik, welche in den Stromnetzen verbaut ist, hat der Netzbetreiber die Möglichkeit, in den Verbrauch der Kunden einzugreifen. Dies kann z.B. bei Gewerbekunden passieren. Hier kann der Netzbetreiber mit dem Gewerbekunden eine Vereinbarung treffen, dass der Netzbetreiber dem Gewerbekunden im Fehlerfall den Strom abstellen darf [6]. Da Gewerbekunden meist Großabnehmer sind, kann dadurch eine Überlastung des Netzes häufig verhindert werden. Wenn solche Vereinbarungen getroffen werden, kann dies in die Netzplanung einfließen. Dann können nämlich die Leitungen im Normalfall stärker ausgelastet werden, da nicht so viel Puffer für den Fehlerfall freigehalten werden muss. Durch den geringeren Pufferbedarf können günstigere Leitungen verwendet werden und somit Kosten gespart werden. Eine weitere Steuerungsmöglichkeit ist das Drosseln von Einspeisungen [6]. Wenn z.B. am Mittag die Solaranlagen ihre maximale Einspeisung haben, es gleichzeitig stark windet und die Windkraftanlagen ebenfalls viel Energie einspeisen, kann dies das Netz überlasten. Um diesen Fall vorzubeugen, muss der Netzbetreiber entweder

mehr Leitungen bzw. Leitungen größerer Kapazität verlegen oder er muss Einspeisungen abregeln dürfen. Diese politische Entscheidung steht nicht wirklich in der Macht der Netzbetreiber. Wenn es jedoch zum Planungszeitpunkt ein Gesetz gibt, welches hinreichend Sicherheit für die Zukunft gibt, kann dies ebenfalls in der Planung berücksichtigt werden. Eine weitere Option ist die Nutzung von Elektroautos als Speicher [6]. Wenn z.B. nachts kein Wind weht, produzieren weder Solaranlagen noch Windkraftwerke Strom. Um dann trotzdem genügend Energie zu haben, könnte die Energie von den Akkus der E-Autos verwendet werden. Hierfür müsste es ein Gesetz geben, welches das regelt und gleichzeitig genügend Anreize für die Kunden, ihr E-Auto zur Verfügung zu stellen. Wenn diese Option jedoch besteht und die Verbreitung von E-Autos etwas größer ist, so kann auch dies in der Planung berücksichtigt werden.

Eine andere Steuerungsstrategie besteht darin, den Strompreis variabel zu machen [6]. Der Strompreis wäre damit günstiger, wenn gerade ein großes Angebot an Strom zur Verfügung steht, z.B. weil die Sonne scheint und es kräftig windet. Andererseits wäre er teurer, wenn gerade wenig Energie produziert wird, bzw. auf Energie von Kohlekraftwerken eine CO_2 Abgabe bezahlt werden muss. Dadurch können die Kunden entscheiden, wann sie den Strom nutzen möchten. Häufig ist es dem Kunden egal, wann sein E-Auto über Nacht aufgeladen wird, solange es am nächsten Morgen genügend Energie hat. Auch in der Industrie gibt es einige Aufgaben, welche erledigt werden müssen, der genaue Ausführungszeitpunkt jedoch ohne Probleme um einige Stunden nach vorne oder hinten verschoben werden kann. In diesen Fällen würden die Kunden, welche möglichst wenig bezahlen wollen, von alleine einer Überlastung entgegenwirken: Wenn zu viel Energie vorhanden ist und diese irgendwie weg muss, können planbare Verbraucher eingeschaltet werden und wenn gerade wenig Energie vorhanden ist und die Spannung einzubrechen droht, ist der Strompreis sehr teuer und alle planbaren Verbraucher schalten sich nicht ein. Mithilfe von Smart Home Geräten und dem Internet der Dinge lässt sich diese Preisregulierung auch komplett automatisch durchführen, wodurch man sich relativ gut darauf verlassen könnte. Auch hierfür muss natürlich eine gesetzliche Regelung mit genügend Sicherheit für die Zukunft bestehen, damit diese Steuerungsstrategie in der Netzplanung berücksichtigt werden kann.

3. Ameisen Algorithmen

In diesem Kapitel wird die Heuristik der Ameisen Algorithmen erklärt. Hierzu wird in Abschnitt 3.1 die allgemeine Idee erläutert, welche aus der Tierwelt stammt. Diese Idee wird abstrahiert und erweitert und in Abschnitt 3.2 genauer erklärt. Hierbei wird der Algorithmus zuerst anhand des Traveling Salesman Problems erklärt, da in diesem die Idee der kürzesten Wege sehr gut übertragbar ist. Anschließend wird die Idee der kürzesten Wege weiter abstrahiert, um schließlich auf beliebige Probleme angewendet werden zu können. In Abschnitt 3.3 beschreibe ich schließlich die Variante Ant Colony System, welche in dieser Bachelorarbeit verwendet wird.

3.1. Die Idee der Ameisenalgorithmen

Die folgende Einführung in Ameisenalgorithmen ist angelehnt an die Art und Weise, wie auch Dorigo und Stützle in Ant Colony Optimization [8] das Thema einführen. Die Einführung basiert zudem auf einem Experiment, welches an Kolonien der argentinischen Ameise durchgeführt wurde [10].

Während sich Ameisen bewegen, hinterlassen sie Duftstoffe, sogenannte Pheromone. Diese können von den anderen Ameisen wahrgenommen werden und beeinflussen deren Entscheidung, wie sie sich fortbewegen. Je größer die Pheromonkonzentration auf einem Weg ist, desto größer ist die Wahrscheinlichkeit, dass die Ameisen diesem Weg folgen. Wenn eine Ameise an einer Kreuzung mehrere Wege W zur Auswahl hat, auf denen jeweils eine gewisse Konzentration Pheromone τ_i vorhanden ist, so wählt die Ameise den Weg w_i mit der Wahrscheinlichkeit

$$P(w_i) = \frac{\tau_i}{\sum\limits_{w_k \in W} \tau_k}. \tag{3.1}$$

Die Wahrscheinlichkeit, dass die Ameise den Weg w_i nimmt, ist also größer, wenn die Pheromonkonzentration τ_i auf dem Weg w_i groß ist. Die Ameisen treffen also eine zufällige Entscheidung, wobei die Wahrscheinlichkeitsverteilung durch die Pheromone beeinflusst wird.

Abbildung 3.1 zeigt ein Beispiel für das Verhalten der Ameisen. Zunächst sind noch auf keinem Weg Pheromone vorhanden und die Ameisen wählen deshalb einen zufälligen Weg. In diesem Beispiel wählt eine Ameisen den oberen, kürzeren, Weg

und eine andere Ameise den unteren, längeren, Weg (siehe Abbildung 3.1a). In diesem Beispiel legt eine Ameise während eines Zeitschritts genau eine Längeneinheit zurück. Der kürzere Weg ist hierbei genau eine Längeneinheit lang, während der längere Weg dagegen zwei Längeneinheiten lang ist.

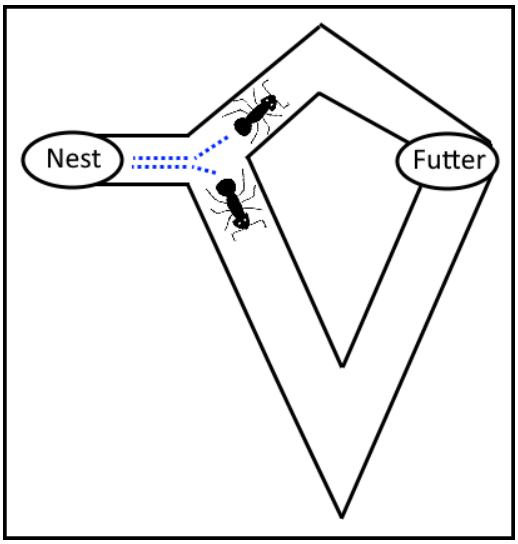
Während jedem Zeitschritt hinterlässt die Ameise eine Einheit Pheromone. Auf dem unteren Weg, wird also pro Überquerung die doppelte Anzahl Pheromone hinterlassen. Da der Weg aber doppelt so lang ist, ist die Konzentration, also die Anzahl der Pheromone pro Längeneinheit, gleich groß. Die Konzentration der Pheromone ist der entscheidende Faktor, nach welchem die Ameisen den Weg wählen. Nach einem Zeitschritt ist die obere Ameise also am Futter angekommen, während die untere Ameise erst bei der Hälfte des Weges ist (siehe Abbildung 3.1b). Die obere Ameise wählt den Heimweg nun nach Formel Gleichung 3.1. Da auf dem unteren Weg noch keine Pheromone sind (und die Wahrscheinlichkeit diesen Weg zu wählen somit 0 beträgt, wenn es einen anderen Weg mit Pheromonen gibt), läuft die Ameise über den oberen Weg wieder zurück. Ein Zeitschritt später ist die Anzahl der Pheromone auf dem oberen Weg also bereits bei zwei, während die Anzahl der Pheromone auf dem unteren Weg erst bei eins ist, da die untere Ameise nun erst beim Futter angekommen ist (siehe Abbildung 3.1c). Die Ameise, welche den unteren Weg gewählt hat, ist nun also am Futter angekommen und muss sich für einen der Wege $W = \{w_o, w_u\}$ entscheiden. Der obere Weg w_o wird hierbei mit der Wahrscheinlichkeit

$$P(w_o) = \frac{\tau_o}{\sum\limits_{w_k \in W} \tau_k} = \frac{\tau_o}{\tau_o + \tau_u} = \frac{2}{2+1} = \frac{2}{3} = 66,\overline{6}\ \% \tag{3.2}$$

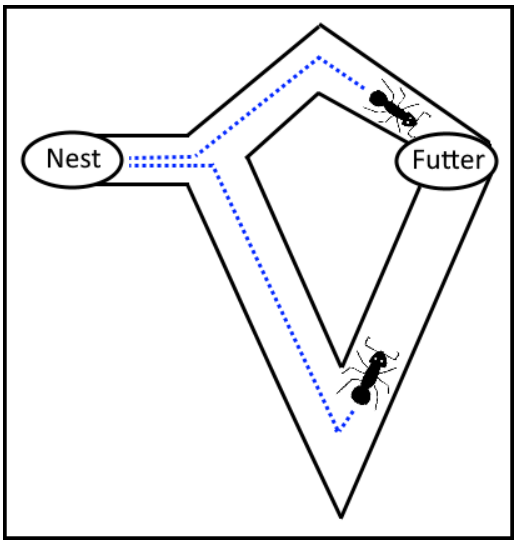
gewählt. Der untere Weg wird mit Wahrscheinlichkeit

$$P(w_u) = \frac{\tau_u}{\sum\limits_{w_k \in W} \tau_k} = \frac{1}{2+1} = \frac{1}{3} = 33,\overline{3}\ \% \tag{3.3}$$

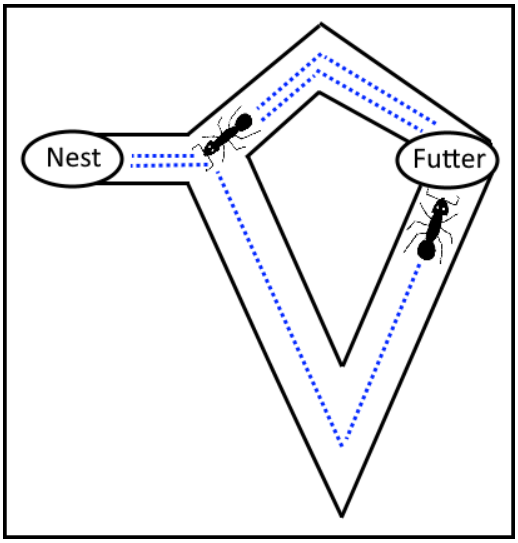
gewählt. In diesem Fall wählt die Ameise den oberen Weg. Währenddessen kommen vom Nest drei neue Ameisen, für die die gleiche Wahrscheinlichkeit gilt, den oberen Weg zu wählen wie in Gleichung 3.2. Die Wahrscheinlichkeit den unteren Weg w_u zu wählen, ist die gleiche wie in Gleichung 3.3. Im Erwartungswert gehen also zwei Ameisen den oberen Weg zum Futter, während eine Ameise den unteren Weg zum Futter wählt (siehe Abbildung 3.1d). Insgesamt gehen also bereits jetzt mehr Ameisen den kürzeren Weg entlang. Dadurch, dass mehr Ameisen diesen Weg schneller überqueren, sammeln sich dort die Pheromone viel schneller an. Das Verhalten der Ameisen konvergiert also dazu, den oberen, kürzeren Weg zu nehmen. Dieses Verhalten wird in der Heuristik der Ant Colony Optimization Algorithmen abstrahiert und erweitert. Dies wird im nächsten Abschnitt genauer erläutert.



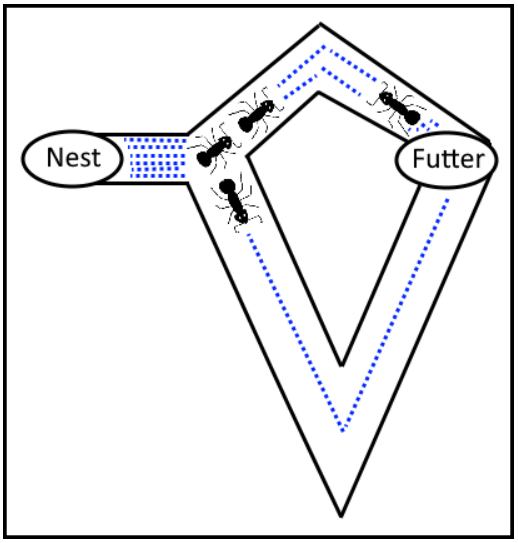
(a) Startsituation, auf keinem der Wege sind Pheromone und die Ameisen wählen den Weg zufällig.



(b) Der obere Weg ist kürzer, wodurch die obere Ameise schneller am Futter ankommt.



(c) Als sich die obere Ameise auf den Weg zum Nest begeben hat war die untere noch nicht am Futter angekommen. Dadurch hatte nur einer der Wege eine Pheromonspur, weshalb die Ameise diesen gewählt hat.



(d) Die Anzahl der Pheromone auf dem oberen Weg beträgt 2, auf dem unteren 1. Die Ameisen wählen deshalb den oberen Weg mit einer größeren Wahrscheinlichkeit (66,6 % gegenüber 33,3 %.

Abbildung 3.1.: Die Ameisen finden den kürzesten Weg zum Futter, indem sie beim laufen Pheromone hinterlassen und diese sich auf kürzeren Wegen schneller ansammeln. Die Pheromone sind hier als gestrichelte Linie symbolisiert. Quelle: Eigene Darstellung

3.2. Ameisenalgorithmen als Heuristik

Im Abschnitt 3.1 wurde erklärt, wie Ameisen kürzeste Wege finden. Hierbei probieren sie zufällig die Wege aus und dadurch, dass sich auf kurzen Wegen die Pheromonkonzentration schneller erhöht als auf langen Wegen, verwenden kurze Zeit später fast alle Ameisen diesen kurzen Weg.

Zunächst betrachten wir jedoch das Travelling Salesman Problem, an welchem ich die Ameisenalgorithmen erklären möchte. In diesem ist ein Graph gegeben. Die Knoten repräsentieren Städte und die Kanten dazwischen sind Straßen, welche die Städte miteinander verbinden. Das Ziel ist es nun, einen möglichst kurzen Rundweg zu finden, welcher alle Städte besucht und wieder in der Ausgangsstadt endet. Nun beginnt man mit einer Kolonie von z.B. zehn virtuellen Ameisen. In der ersten Iteration konstruiert nun jede Ameise eine vollkommen zufällige Lösung. Nachdem alle Ameisen ihre Lösung konstruiert haben, wird jede Lösung bewertet. Je kürzer die konstruierte Lösung, desto mehr Pheromone werden auf den jeweiligen Komponenten der Lösung, in diesem Fall also den Kanten, hinzugefügt. Dies simuliert, dass die Ameisen aufgrund der kürzeren Strecke schneller sind und sich auf diesen Wegen die Pheromonkonzentration somit schneller erhöht. Eine Art, die Menge an Pheromonen zu bestimmen, welche hinzugefügt wird, ist z.B. die inverse Weglänge. Wenn der Weg von Ameise eins z.B. zehn Längeneinheiten lang ist, so erhalten alle beteiligten Komponenten $\frac{1}{10}$ an Pheromonen hinzu. Nachdem alle Ameisen ihre Pheromone auf den Kanten verteilt haben, beginnt die nächste Iteration. Nun haben einige Kanten mehr Pheromone als andere und somit wandern die Ameisen nicht mehr komplett zufällig auf dem Graphen umher, sondern werden von den Pheromonen beeinflusst. Hierbei findet erneut Gleichung 3.1 Verwendung. Eine Kante, welche in vielen guten Lösungen enthalten ist, erhält schneller Pheromone und wird somit mit großer Wahrscheinlichkeit in zukünftigen Iterationen erneut gewählt. Dies repräsentiert das Ausnutzen der Pheromonkonzentration, mit welcher die Ameisen kurze Wege finden.

Diese erste Version ist jedoch noch nicht so erfolgreich, da die zufälligen Entscheidungen der Ameisen in den ersten Iterationen einen zu großen Einfluss haben. So können z.B. Komponenten, welche in der ersten Iteration in keiner Lösung enthalten sind, nie wieder gewählt werden, da sie null Pheromone haben und die Wahrscheinlichkeit gewählt zu werden somit null beträgt (es sei denn, dies wäre die einzige noch gültige Option). Die meisten Versionen geben deshalb allen Kanten eine Startkonzentration an Pheromonen. Dadurch sind in der ersten Iteration immer noch alle Möglichkeiten gleich wahrscheinlich, aber der Einfluss der ersten Iteration auf den folgenden Suchprozess ist geringer. Besonders am Anfang, wenn die Pheromone noch nicht so gut verteilt sind, erhalten auch Komponenten Pheromone, obwohl diese eigentlich nie an guten Lösungen beteiligt sind. Um diesen Einfluss wieder loszuwerden, verwenden viele Varianten der Ameisenalgorithmen einen sogenannten Verdunstungsschritt. In diesem werden allen Kanten z.B. 5 % ihrer Pheromone entfernt. Komponenten, die stets an guten Lösungen beteiligt sind, erhalten diese Pheromone und weitere schnell zurück. Komponenten, die hingegen nur durch Zufall

Pheromone erhalten haben, verlieren ihren Vorteil schnell wieder. So wird gewährleistet, dass nur die Komponenten, welche regelmäßig an guten Lösungen beteiligt sind, eine große Pheromonkonzentration aufrecht erhalten können.

Um den Suchprozess am Anfang zu beschleunigen, kann man die Ameisen nicht nur nach den Pheromonwerten Entscheidungen treffen lassen, sondern zudem noch nach sogenannten heuristischen Werten η. Dies können im Fall des Traveling Salesman Problem z.B. die inverse Distanz sein (da größere Werte besser sind, wird die inverse Weglänge gewählt, um kurzen Strecken höhere Werte zu geben als längeren). Der Grund dafür ist, dass eine optimale Route meist aus vielen kurzen Kanten und nur wenigen langen Kanten besteht [8]. Somit gibt man den Ameisen ein wenig Vorwissen mit. Die Ameisen treffen ihre Entscheidung dann basierend auf den heuristischen Werten (unserem Vorwissen) sowie ihrem eigenen gesammelten Wissen (den Pheromonen). Hierbei kann man mithilfe von Parametern zudem bestimmen, wie die Gewichtung der Pheromonwerte im Vergleich zu den heuristischen Werten stattfinden soll. Insgesamt ergibt sich also folgende Formel:

$$P(Z = w_i) = \frac{\tau_i^\alpha \cdot \eta_i^\beta}{\sum\limits_{w_k \in W} \tau_k^\alpha \cdot \eta_k^\beta}. \tag{3.4}$$

Hierbei ist Z eine Zufallsvariable, welche die Werte von allen gültigen Wegen W annehmen kann. Die Wahrscheinlichkeit dafür, dass die Ameise den Weg w_i wählt (also $Z = w_i$ gilt) setzt sich also folgendermaßen zusammen: Je größer die Pheromonkonzentration τ_i ist und je größer der heuristische Wert η_i, desto wahrscheinlicher wählt die Ameise den Weg w_i. Mithilfe der Parameter α und β kann man zudem noch gewichten, wie groß die Rolle der Pheromone und der heuristischen Werte ist. Wenn α oder β den Wert Null annehmen, so haben entweder die Pheromone oder die heuristischen Werte keinen Einfluss auf das Ergebnis. Im Nenner wird dasselbe Produkt für alle Wege, welche die Ameise in diesem Schritt gehen kann, gebildet und die einzelnen Produkte werden aufsummiert. Auf diese Art und Weise ist gewährleistet, dass $\sum_{w_k \in W} P(Z = w_k) = 1$ gilt. Da der Nenner für alle Wegoptionen $w_k \in W$ gleich ist, hat nur der Zähler einen Einfluss auf die Wahrscheinlichkeit, dass dieser Weg gewählt wird. Im Fall des Traveling Salesman Problem ist die Menge W die Menge aller Städte, welche von der aktuellen Stadt erreicht werden können. Diese Menge ändert sich natürlich, sobald die Stadt gewechselt wurde. Hinzu kommt, dass alle Städte, die bereits besichtigt wurden, ebenfalls aus der Menge W herausgenommen werden. Dies kann natürlich zu der Situation führen, dass die Menge W leer ist, obwohl noch nicht alle Städte besucht wurden. In diesem Fall hat man zwei Optionen. Entweder startet man die Lösungskonstruktion von vorne oder man lockert die Bedingung, z.B. dass in diesem Schritt auch bereits besuchte Städte erneut besucht werden dürfen. Wenn man sich für die zweite Methode entscheidet, kann man den Kosten der Lösung noch einen Strafterm hinzufügen, wenn eine Randbedingung verletzt wurde.

3.3. Ant Colony System

Die Variante, welche in dieser Arbeit verwendet wird, nennt sich Ant Colony System. Diese Variante wurde gewählt, da sie bei den bisherigen Problemen (Traveling Salesman Problem, Network Routing, Scheduling, Subset, ...) zu den besten Ergebnissen geführt hat [8]. Zudem wurde diese Variante bereits erfolgreich auf die Netzplanung angewendet. Lukas Gebhard konnte in seiner Masterarbeit zeigen, dass die Niederspannungsnetzplanung mithilfe von Ant Colony System erfolgreich funktioniert [11]. Deshalb wird diese Variante nun auch in dieser Arbeit für die Planung von Mittelspannungsnetzen verwendet. Ant Colony System hat im Vergleich zu den anderen Varianten einige Änderungen, welche ich im folgenden kurz erläutern möchte.

3.3.1. Auswahl der Komponenten

Die erste Änderung im Vergleich zu den bisherigen Varianten besteht darin, dass die Formel, nach welcher die Ameisen ihre Entscheidung treffen, anders ist. Die Ameisen entscheiden hierbei nach folgender Formel:

$$w = \begin{cases} w_i \ , \ i = \underset{i}{\operatorname{argmax}} \ \{\tau_i^\alpha \cdot \eta_i^\beta\} & \text{wenn } q \leq q_0 \\ Z & \text{sonst} \end{cases} . \tag{3.5}$$

Dabei ist w der Weg, welchen die Ameise wählt. Dieser hängt maßgeblich von q, einer Zufallsvariablen, ab. q ist gleichverteilt im Intervall $[0, 1]$ und q_0 ist eine Konstante zwischen 0 und 1. q_0 gibt an, wie groß die Wahrscheinlichkeit ist, dass die beste, nach dem Wissen der Pheromone und der Heuristik, zur Verfügung stehende Option gewählt wird. Wenn jedoch $q > q_0$ ist, so wird das Ergebnis der Zufallsvariablen Z benutzt. Die Wahrscheinlichkeitsverteilung für die Zufallsvariable Z ist die gleiche wie in Gleichung 3.4. In dieser Variante entscheiden sich die Ameisen also mit Wahrscheinlichkeit q_0 für die nach ihrem Wissen beste Option und nur mit der Wahrscheinlichkeit $1 - q_0$ wird die Wahl des Weges dem Zufall überlassen, wie es bei den üblichen Ameisenalgorithmen der Fall ist.

Eine weitere Eigenschaft, welche Ant Colony System von den üblichen Ameisenalgorithmen unterscheidet, ist die Art und Weise, wie das Pheromonupdate durchgeführt wird. Hierbei hinterlässt nicht jede Ameise Pheromone, sondern nur die beste Ameise. Dies kann je nach Implementierung die beste Ameise der aktuellen Iteration sein oder die beste seit Beginn der Optimierung. In letzterem Fall kann es durchaus vorkommen, dass über mehrere Iterationen keine Ameise der jeweils aktuellen Iteration Pheromone verteilt. Auch die Verdunstung der Pheromone findet nur auf dieser besten Lösung statt. Die Gleichung

$$\tau_i \leftarrow (1 - \rho) \cdot \tau_i + \rho \cdot \Delta\tau_i \qquad \forall \tau_i \in T^{bs} \tag{3.6}$$

beschreibt hierbei, wie das Pheromonupdate durchgeführt wird. ρ ist ein Parameter, welcher im Intervall $[0, 1]$ enthalten ist. $(1 - \rho)$ ist der Parameter, welcher für die Verdunstung zuständig ist. ρ ist der Parameter, mit welchem die Menge der hinzugefügten Pheromone beeinflusst wird. $\Delta\tau_i$ ist die ermittelte Pheromonkonzentration, welche für die Komponenten der Lösung hinzugefügt werden soll. Im Beispiel des Traveling Salesman Problem kann $\Delta\tau_i$ z.B. $\frac{1}{\text{Weglänge}}$ sein. Dieses Update wird für alle Komponenten durchgeführt, welche an der besten Lösung T^{bs} beteiligt sind. Wie schon erwähnt, kann dies entweder die beste Lösung der aktuellen Iteration sein oder die beste Lösung seit Beginn der Suche. Die Art des Updates kann man zudem als eine Art gewichtete Mittelwertbildung sehen, wobei ρ der Faktor für die Mittelwertbildung ist. Im Fall des Traveling Salesman Problem gibt es eine optimale Lösung, welche eine Weglänge l hat. Somit gilt $\Delta\tau_i \leq \frac{1}{l}$, die Pheromonkonzentration ist also nach oben beschränkt.

3.3.2. Der Verdunstungsschritt

Zudem gibt es noch eine weitere Art der Verdunstung. Diese findet jedoch ebenfalls auf eine andere Art und Weise statt. Die übliche Verdunstung, in welcher in jeder Iteration allen Kanten ein gewisser Prozentwert abgezogen wird, findet hier nicht statt. Stattdessen entfernen die Ameisen bei ihrer Lösungskonstruktion Pheromone auf den Kanten, die sie benutzen. Dieser Schritt soll dazu führen, dass die anderen Ameisen eher neue Komponenten ausprobieren und nicht dieselben Komponenten benutzen. Dadurch, dass die Ameisen ihre Entscheidung, welchen Weg sie wählen, nach Gleichung 3.5 treffen, ist die Wahrscheinlichkeit trotzdem hoch, dass sehr gute Komponenten mit vielen Pheromonen trotzdem weiterhin gewählt werden. Dies liegt daran, dass die besten Komponenten mit Wahrscheinlichkeit q_0 gewählt werden. Durch eine kleine Reduzierung der Pheromone bleibt die beste Komponente meistens trotzdem weiterhin die beste Komponente, weshalb sie dann erneut mit Wahrscheinlichkeit q_0 in der nächsten Lösung enthalten ist. Nur im Fall, dass q einen Wert größer als q_0 annimmt, ist die Wahrscheinlichkeit niedriger, dass diese gute Komponenten weiterhin gewählt werden. Für weniger gute Komponenten gilt dieses Argument jedoch nicht. Diese werden durch die Reduzierung seltener gewählt, was einerseits zu einer größeren Erkundung führt und andererseits dazu führt, dass schlechtere Komponenten in Zukunft seltener gewählt werden. Die Formel mit welcher dieses lokale Pheromonupdate berechnet wird, ist die Folgende:

$$\tau_i \leftarrow (1 - \xi) \cdot \tau_i + \xi \cdot \tau_0. \tag{3.7}$$

Hierbei ist $(1 - \xi)$ der Verdunstungsparameter und ξ ist der Parameter, welcher mit der Anfangskonzentration der Pheromone multipliziert wird. Auch hierbei kann man das Pheromonupdate wieder als Bildung eines gewichteten Mittelwertes sehen. In diesem Fall sorgt das dafür, dass die Pheromonkonzentration auf keiner Kante

unter das Ausgangsniveau τ_0 fallen kann. Dies gilt, sofern das globale Pheromonupdate die Konzentration stets größer als τ_0 hält. Dies ist z.B. dadurch möglich, dass man beim globalen Pheromonupdate die beste bisher gefundene Lösung benutzt. Dann kann man eine Ameise eine Lösung konstruieren lassen und findet $\Delta\tau_i$ für die Komponenten der Lösung heraus. Wenn man nun

$$\tau_0 = \frac{\Delta\tau_i}{x} , \qquad x > 1$$

wählt, so wird im globalen Update $\Delta\tau_i$ stets größer als τ_0 sein, wodurch gilt, dass τ_0 eine Untergrenze für die Pheromonkonzentration ist.

Algorithmus 1 zeigt den Pseudocode für Ant Colony System. Zu Beginn des Algorithmus müssen alle Daten initialisiert werden. Dazu gehört insbesondere das Erstellen der Datenstruktur, auf welcher alle Ameisen umherwandern, das Ausrechnen der heuristischen Werte, sowie die Verteilung der Anfangskonzentration τ_0. Im nächsten Schritt kann die Optimierungsschleife losgehen. In jeder Iteration wird zunächst eine leere Liste erzeugt, in welcher die Lösungen aller Ameisen gespeichert werden. Anschließend wird über jede Ameise iteriert. Zunächst erstellt die Ameise eine Lösung. Im Anschluss daran können optional noch lokale Optimierungen an der Lösung vorgenommen werden. Ein Beispiel für lokale Optimierungen im Traveling Salesman Problem ist in Abbildung 3.2 zu sehen. Hierbei wird für jede Kante geprüft, ob es nicht effizienter wäre, die benachbarten Kanten an die anderen Endpunkte der eigenen Kante anzuschließen. Dies entspräche dem Ändern der Route, so dass die beiden Städte der betrachteten Kante die Besuchsreihenfolge tauschen. Nach dem Postprocessing wird das lokale Pheromonupdate durchgeführt, damit die anderen Ameisen mehr zum Erkunden angeregt werden. Nachdem die nun fertigen Lösungen zur Liste aller Lösungen hinzugefügt wurde, muss nur noch die Statistik aktualisiert werden. Die Statistik enthält wichtige Informationen, z.B. wie die beste Lösung aussieht, welche Kosten die beste Lösung hatte etc. Sie kann jedoch auch andere Informationen enthalten, wie z.B. die benötigte Zeit pro Iteration, welche für den Benutzer interessant sein können, für den Algorithmus jedoch nicht wichtig sind. Nachdem dieser Prozess für alle Ameisen durchgeführt wurde, findet das globale Pheromonupdate statt. Hierbei findet die Verdunstung sowie das Hinzufügen von Pheromonen für die Komponenten der besten Lösung statt. Nach all diesen Schritten kann die nächste Iteration beginnen. Am Ende von allen Iterationen befindet sich in der Statistik die beste gefundene Lösung.

Auf die Implementierung der Methoden construct_solution und postprocessing wird in diesem Kapitel nicht eingegangen, da diese problemspezifisch sind. Diese Methoden werden im Traveling Salesman Problem anders implementiert als in der Netzplanung. Kapitel 4 beschreibt die Implementierung dieser Methoden für den konkreten Fall der Mittelspannungsnetzplanung.

Zuletzt möchte ich noch eine Bemerkung zur Lösung von anderen Problemen machen. Die Art der Lösungssuche kann nämlich einfach abstrahiert werden, wodurch

Algorithm 1 Ant Colony System

```
function ACS_SEARCH(nr_of_iterations)
    initialize_data()
    for iteration = 0 to  nr_of_iterations − 1 do
        solutions = []                                   ▷ create empty list
        for ant_nr = 0 to  nr_of_ants − 1 do
            solution ← ants[ant_nr].construct_solution()
            solution ← postprocessing(solution)          ▷ optional
            local_pheromone_update(solution)
            solutions.add(solution)
            update_statistics(solution)
        end for
        global_pheromone_update(solutions)
    end for
end function
```

nicht nur Probleme, in welchen ein kürzester Weg gefunden werden soll, gelöst werden können. Hierzu wählt man eine Größe des Problems aus, welche minimiert werden soll. Dann erzeugen die Ameisen jeweils ihre Lösungen und je kleiner die Kosten der Lösung sind, desto mehr Pheromone erhalten die Komponenten der Lösung, bzw. desto größer ist $\Delta\tau_i$ in der Ant Colony System Version.

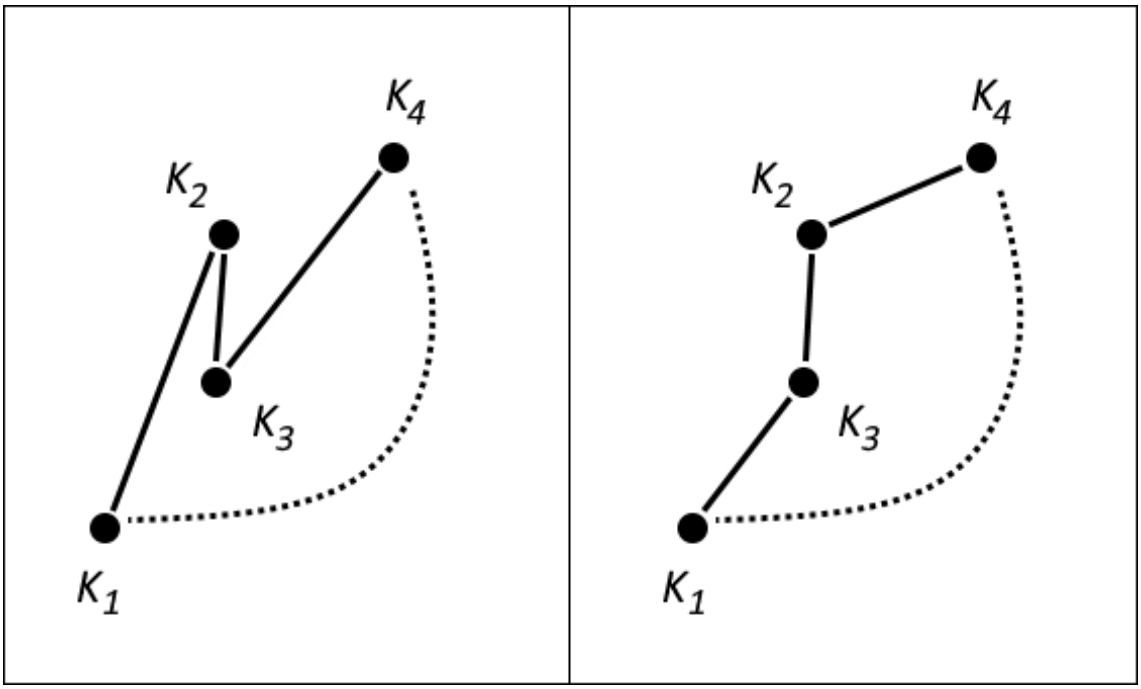


Abbildung 3.2.: Diese Abbildung zeigt eine Möglichkeit für eine lokale Optimierung im Traveling Salesman Problem. In diesem Beispiel wird die lokale Optimierung für die Kante von K_2 nach K_3 durchgeführt. Auf der linken Seite sieht man die Ausgangssituation, in welcher die Besuchsreihenfolge $(K_1, K_2, K_3, K_4, ...)$ ist. Durch Tauschen der Besuchsreihenfolge zu $(K_1, K_3, K_2, K_4, ...)$ kann in manchen Fällen eine Optimierung entstehen. Wenn diese neue Route besser ist, so wird sie beibehalten, ansonsten wird die Änderung wieder verworfen. Diese lokale Optimierung wird nun für jede Kante der Lösung durchgeführt. Quelle: Eigene Darstellung

4. Der Algorithmus

In diesem Kapitel beschreibe ich im Detail, wie der Algorithmus funktioniert. Der Algorithmus wurde in Python implementiert. Hierbei wurden einige Software Bibliotheken bzw. Software Toolboxes verwendet. Einerseits PyPSA [12] für die Modellierung von elektrischen Netzen. Andererseits SciPy [13] für die Berechnung der Triangulierung (siehe Abschnitt 4.3). NetworkX [14] wurde für Graphen und Graphenalgorithmen benötigt und für die Visualisierungen wurde Matplotlib [15] eingesetzt. Das Speichern und Laden von Daten wurde mit PyYAML [16] realisiert.

4.1. Idee des Algorithmus

In diesem Kapitel wird die Grundidee des Algorithmus erklärt. Die einzelnen Schritte werden in den folgenden Kapiteln jeweils ausführlich erklärt.

Um in einem Stromnetz alle Verbraucher und Erzeuger miteinander zu verbinden reicht es, alle Busse miteinander zu verbinden. Dies liegt daran, dass Verbraucher, Erzeuger und Speicher sowie Leitungen und Schalter alle an Busse angeschlossen sind. Deshalb wird in der folgenden Planung nur auf die Verbindung der Busse untereinander eingegangen. Zu Beginn erhält der Algorithmus eine Eingabe, welche die Busse und ihre Koordinaten enthält. Auf dieser findet nun eine Triangulierung statt. Die Triangulierung erzeugt aus einer Punktmenge Dreiecke, deren Ecken in der Punktmenge enthalten sind und sich nicht überschneiden. Durch die Triangulierung wird der Suchraum verkleinert und zudem sichergestellt, dass nur gültige Ringe gebildet werden. Die Busse sind nun die Eckpunkte der Dreiecke. Nun kann eine Ameise mit der Optimierung beginnen. Hierzu wählt sie ein Dreieck aus, welches einen Bus enthält, der mit einem Transformator verbunden ist. Von nun an ist das Stromnetz in einem gültigen Zustand. Es wurden zwar noch nicht alle Busse verbunden, aber das Stromnetz erfüllt alle Kriterien, welche ein Mittelspannungsnetz erfüllen muss (siehe Abschnitt 2.2). In der Lösungskonstruktion der Ameise wird von nun garantiert, dass das Mittelspannungsnetz nach jedem Hinzufügen eines Busses weiterhin in einem gültigen Zustand ist. Auf diese Art und Weise wird nun nach und nach jeder Bus mit dem Mittelspannungsnetz verbunden. Ein Beispiel für eine solche Lösungskonstruktion ist in Abbildung 4.2 in Abschnitt 4.5 zu finden (Kacheln 3 bis 8). Bei der Erweiterung wird hierbei eine Kante entfernt, damit das Netz weiterhin in einem gültigen Zustand verbleibt. Nachdem die Ameise auf diese Art und Weise alle Busse verbunden hat, finden in einem optionalen Schritt einige

lokale Optimierungen statt. Eine dieser lokalen Optimierungen ist die Stichoptimierung (siehe Abbildung 4.4 in Abschnitt 4.6). Nach den lokalen Optimierungen wird die Lösung bewertet, ggf. in der Statistik vermerkt und das lokale Pheromonupdate wird durchgeführt. Nun kommen die anderen Ameisen dran und erzeugen ihre Lösung. Wenn alle Ameisen fertig sind, findet das globale Pheromonupdate statt. Im Anschluss darauf beginnt die nächste Iteration und die Ameisen fangen wieder von vorne mit dem Aufbau einer Lösung an, dieses Mal jedoch mit etwas mehr Wissen, in Form von unterschiedlichen Pheromonwerten auf den Kanten des Graphen.

4.2. Eingabe und Parameter

In diesem Teil des Kapitels geht es darum, welche Eingaben der Algorithmus benötigt und welche Parameter eingestellt werden können, um das Verhalten des Algorithmus zu beeinflussen.

4.2.1. Eingabe

Als Eingabe wird lediglich ein PyPSA [12] Netz benötigt. PyPSA ist eine Software Toolbox, welche für die Modellierung und Berechnung von elektrischen Netzen verwendet wird. Das Stromnetz muss die Busse enthalten. Diese müssen zudem Koordinaten haben, da ansonsten die Triangulierung nicht funktioniert. Des Weiteren muss das PyPSA-Netz noch die möglichen Leitungen enthalten, die realisierbar sind, auch wenn diese aktuell noch nicht existieren. Jede Leitung muss zudem eine Länge hinterlegt haben, da diese für die Berechnung der Kosten benötigt wird. In einer zukünftigen Version soll zudem ein zweites PyPSA Netz mit dem aktuell vorhandenen Stromnetz übergeben werden. Dann können die Ameisen bereits existierende Leitungen kostenlos mitbenutzen. Zudem können ebenfalls die Kosten für den Rückbau nicht mehr benötigter Leitungen mit eingerechnet werden.

4.2.2. Parameter

Die wohl wichtigsten Parameter, welche einstellbar sind, sind die Parameter, welche die Ameisen beeinflussen. Hierbei natürlich insbesondere die Parameter aus Gleichung 3.4, Gleichung 3.5, Gleichung 3.6 und Gleichung 3.7. Um die Gewichtung der Pheromone zu bestimmen, kann der Parameter α eingestellt werden, welcher standardmäßig 1 beträgt. Zur Gewichtung der heuristischen Werte kann der Parameter β eingestellt werden, welcher standardmäßig ebenfalls bei 1 liegt. Sowohl α als auch β müssen mindestens null sein. Mithilfe des Parameters q_0 kann entschieden werden, wie groß die Wahrscheinlichkeit ist, dass die aktuell beste Option gewählt wird. Da es sich um eine Wahrscheinlichkeit handelt, muss der Wert zwischen null und eins liegen. Im globalen Update muss zusätzlich der Parameter ρ

eingestellt werden, welcher sowohl die Verdunstung als auch die Menge neu hinzugefügten Pheromone steuert. Der Parameter muss zwischen null und eins liegen. Beträgt er null, so findet kein Pheromonupdate statt. Wenn er hingegen eins beträgt, so ist die Menge der Pheromone nur von der besten Lösung abhängig, an welcher die jeweilige Komponente beteiligt war. Standardmäßig liegt der Wert bei $0,1$. Der Parameter ξ, welcher für das lokale Pheromonupdate zuständig ist, muss ebenfalls zwischen null und eins liegen. Liegt er bei 0, so nehmen die Ameisen keine Pheromone weg, wodurch die anderen Ameisen weniger zum Erkunden angeregt werden. Liegt er hingegen bei 1, so wird jede gewählte Kante wieder auf die Pheromonkonzentration vom Anfang zurückgesetzt, was die anderen Ameisen zu sehr vielen Erkundungen anregen würde. Dies würde jedoch auch verhindern, dass Wissen in Form von Pheromonen gespeichert wird. Auch dieser Parameter liegt standardmäßig bei $0,1$. In der Optimization Loop kann eingestellt werden, ob bereits während jeder Iteration das Postprocessing stattfinden soll, oder ob es erst am Ende einmalig stattfinden soll. Dies ist einerseits eine Frage der Laufzeit, da das Postprocessing natürliche ebenfalls eine Menge Zeit benötigt. Andererseits ist es auch eine Frage der Ergebnisqualität. Des Weiteren kann noch die Anzahl an Ameisen, über den Parameter ants_per_colony, eingestellt werden. Standardmäßig beträgt die Anzahl an Ameisen pro Kolonie 5. Als letztes kann, mithilfe des Parameters iterations, noch die Anzahl an Iterationen eingestellt werden.

Neben den Parametern zum Einstellen des Verhaltens der Ameisen, können natürlich auch noch einige Parameter bezüglich der elektrischen Netze eingestellt werden. So wird eine Schätzung über die minimalen Expansionskosten benötigt, um die Qualität der Lösungen zu bewerten. Anhand dieses Parameters wird dann entschieden, wie viele Pheromone für eine erstellte Lösung vergeben werden. Zudem kann die Option ignore_high_voltage_buses gewählt werden, bei welcher die Busse aus dem Hochspannungsnetz ignoriert werden. In den meisten Netzen sind nämlich welche enthalten, da die Transformatoren, welche das Mittelspannungsnetz versorgen, ihre Energie aus dem Hochspannungsnetz erhalten. Dadurch ist ein Transformator an zwei Busse angeschlossen, an einen aus dem Hochspannungsnetz und an einen aus dem Mittelspannungsnetz. Wird diese Option nicht gewählt, kann der Algorithmus nicht fortfahren, da der Algorithmus in seiner aktuellen Version lediglich Mittelspannungsnetze optimiert.

Zudem sind noch einige Parameter zum Verhalten des Algorithmus vorhanden. So kann mithilfe eines Seeds, die Reproduzierbarkeit der Ergebnisse gewährleistet werden. Zudem kann entschieden werden, ob alle Testnetze optimiert werden, oder nur ein ausgewähltes Netz optimiert wird.

In einer zukünftigen Version wird außerdem ein Parameter für die maximale Leitungsauslastung im Normalfall benötigt. Dieser gibt an, wie viel Puffer für den Fehlerfall freigehalten wird. Je nach gewähltem Steuerungsregime (siehe Abschnitt 2.3) kann dieser unterschiedlich ausfallen. Wenn der Netzbetreiber Großunternehmern im Fehlerfall den Strom abstellen darf und Einspeisungen in Extremsituationen abregeln darf, kann die Leitungsauslastung im Normalfall viel größer sein.

4.3. Triangulierung

Zu Beginn des Algorithmus findet die Triangulierung statt. Bei dieser werden die Koordinaten der Busse von der Eingabe benutzt. Die Busse sind nun die Eckpunkte von Dreiecken. Hierbei werden sie so verbunden, dass nur Dreiecke entstehen und sich keine Kanten überschneiden. In diesem Fall wird die sogenannte Delaunay Triangulation verwendet. Dies liegt daran, dass im Traveling Salesman Problem 99 % der Kanten, welche zur optimalen Lösung gehören, auch in der Delaunay Triangulierung enthalten sind [6]. Auch bei dem Problem der Mittelspannungsnetzplanung ist davon auszugehen, dass dies ähnlich ist, da durch die Triangulierung benachbarte Busse meist verbunden werden und vor allem Kanten zwischen Bussen nicht mehr betrachtet werden, wenn die Busse weit auseinander sind. Aus diesem Grund entscheidet sich Niklas Rotering, ebenfalls die Delaunay Triangulierung für die Netzplanung zu verwenden und ich folge hier seinem Ansatz. In Abbildung 4.2 kann man in der ersten Kachel zudem sehen, wie die Eingabe nach der Triangulierung aussieht. Das Quadrat repräsentiert hierbei den Transformator, die Kreise die Busse und die Kanten dazwischen sind Leitungen, welche von den Ameisen gewählt werden können. Wie man sieht, sind nur benachbarte Busse miteinander verbunden, obwohl theoretisch alle Busse miteinander verbunden werden können. Durch diese Reduktion kann der Suchraum für die Ameisen deutlich verkleinert werden. Im Fall des Traveling Salesmal Problem kann dies ohne nennenswerte Einbußen bei der Qualität der Lösung durchgeführt werden [6]. Ein weiterer wichtiger Grund für die Triangulierung ist, dass dadurch garantiert werden kann, dass die Ameisen stets Ringe bauen und sich somit automatisch an die Kriterien der Versorgungssicherheit halten. Dieser Punkt wird nochmals genauer in Abschnitt 4.5 erläutert. Zur Realisierung der Delauny Triangulation wurde eine Funktion aus SciPy [13] verwendet.

4.4. Grapherzeugung

Bevor die Ameisen das Mittelspannungsnetz nach und nach aufbauen können, benötigen sie erst einmal eine Datenstruktur, auf welcher sie operieren können. Hierbei bietet sich ein Graph an, da die Ameisen auf diesem einfach den Kanten folgen können und auch die Pheromone können auf den Kanten platziert werden. Im folgenden wird deshalb erklärt, wie der Graph erzeugt wird. Zuerst wird auf die Erstellung der Knoten und Kanten eingegangen, bevor dann auch die Ermittlung der Kantengewichte beschrieben wird. Zur Realisierung des Graphen in Python verwende ich die Bibliothek NetworkX [14].

4.4.1. Knoten und Kanten

In Abbildung 4.1 sieht man, wie der Graph erstellt wird. Die erste Kachel zeigt die Ausgangssituation der Erstellung, welche aus der Triangulierung der Busse be-

steht. Jedes Dreieck wird in dem Graphen durch einen Knoten repräsentiert. In der zweiten Kachel ist dies damit visualisiert, dass jeder Knoten (grün) innerhalb des Dreiecks liegt, welches er repräsentiert. Die schwarzen Kreise sind die Busse und die schwarzen Kanten sind die Kanten der Triangulierung. Die Kanten des Graphen (grün) verbinden genau die Knoten miteinander, welche benachbart sind. In diesem Fall zählen zwei Dreiecke genau dann als benachbart, wenn sie sich eine Seite teilen. Dies ist in der dritten Kachel zu sehen. Die Kanten sind in grün visualisiert.

Zudem gibt es noch einen Spezialfall, welcher hier behandelt werden muss. Dadurch, dass die Ameisen die Erzeugung von einem Transformator starten müssen, wird jeder Bus, an welchen ein Transformator angeschlossen ist, ebenfalls als Knoten im Graphen repräsentiert. Dieser Knoten ist mit allen Dreiecken verbunden, in welchen er als Eckpunkt enthalten ist. Dies ist in der vierten Kachel zu sehen, wo der Knoten, welcher den Transformatorbus repräsentiert, als Quadrat markiert wurde. Dieser Knoten ist sowohl Teil der Triangulierung als auch Teil des Graphen. Da die Ameisen natürlich nur eine Erweiterung von dem Transformator starten können, dort jedoch nicht während der Erweiterung hinlaufen dürfen, wird der Graph als gerichteter Graph implementiert. Alle bisherigen Kanten werden zu zwei Kanten mit Hin und Rückweg. Einzige Ausnahme sind die Kanten, von einem Bus, welcher an einen Transformator angeschlossen ist. Dieser hat nur ausgehende Kanten und keine eingehenden Kanten. Eine eingehende Kante würde bedeuten, dass das Mittelspannungsnetz von dem Dreieck aus zum Transformator erweitert wird. Wenn das Dreieck jedoch schon zum Netz gehört, so gehört auch der Transformator bereits zum Netz, da er ein Eckpunkt des Dreiecks ist. Aus diesem Grund kann keine Erweiterung des Netzes zum Transformator stattfinden, weshalb der Knoten, der den Transformator repräsentiert, auch keine eingehenden Kanten hat.

4.4.2. Kantengewichte

Die Kantengewichte repräsentieren die Kosten, welche entstehen, wenn die Ameise das Mittelspannungsnetz über diese Kante vergrößert. Dies ist in Abbildung 4.1 zu sehen. Die Berechnung der Kantengewichte findet hierbei so statt, dass die Leitungen, welche neu hinzukommen, zu den Kosten addiert werden. Die Leitungen, welche eingespart werden, fließen negativ in die Berechnung ein. Zu Beginn der Lösungskonstruktion, wenn noch kein Dreieck zum Netz dazugehört, entstehen die Kosten, welche nötig wären, um alle drei Leitungen des Dreiecks zu bauen, wenn dieses zum Start gewählt werden würde. Da der Netzaufbau an einem Transformator starten muss, gibt es in diesem Minimalbeispiel nur einen möglichen Start. Die Ameisen müssen ihre Optimierung also auf dem Knoten des Transformators starten und als erstes zum Knoten im rechten Dreieck laufen. Um das daraus resultierende Stromnetz zu realisieren, entstehen in diesem Fall Kosten in Höhe von $1+2+3 = 6$. Deshalb beträgt das Kantengewicht der Kante vom Transformator zu dem rechten Dreieck 6. Dies ist in der fünften Kachel der Abbildung zu sehen. Wie bereits im vorherigen

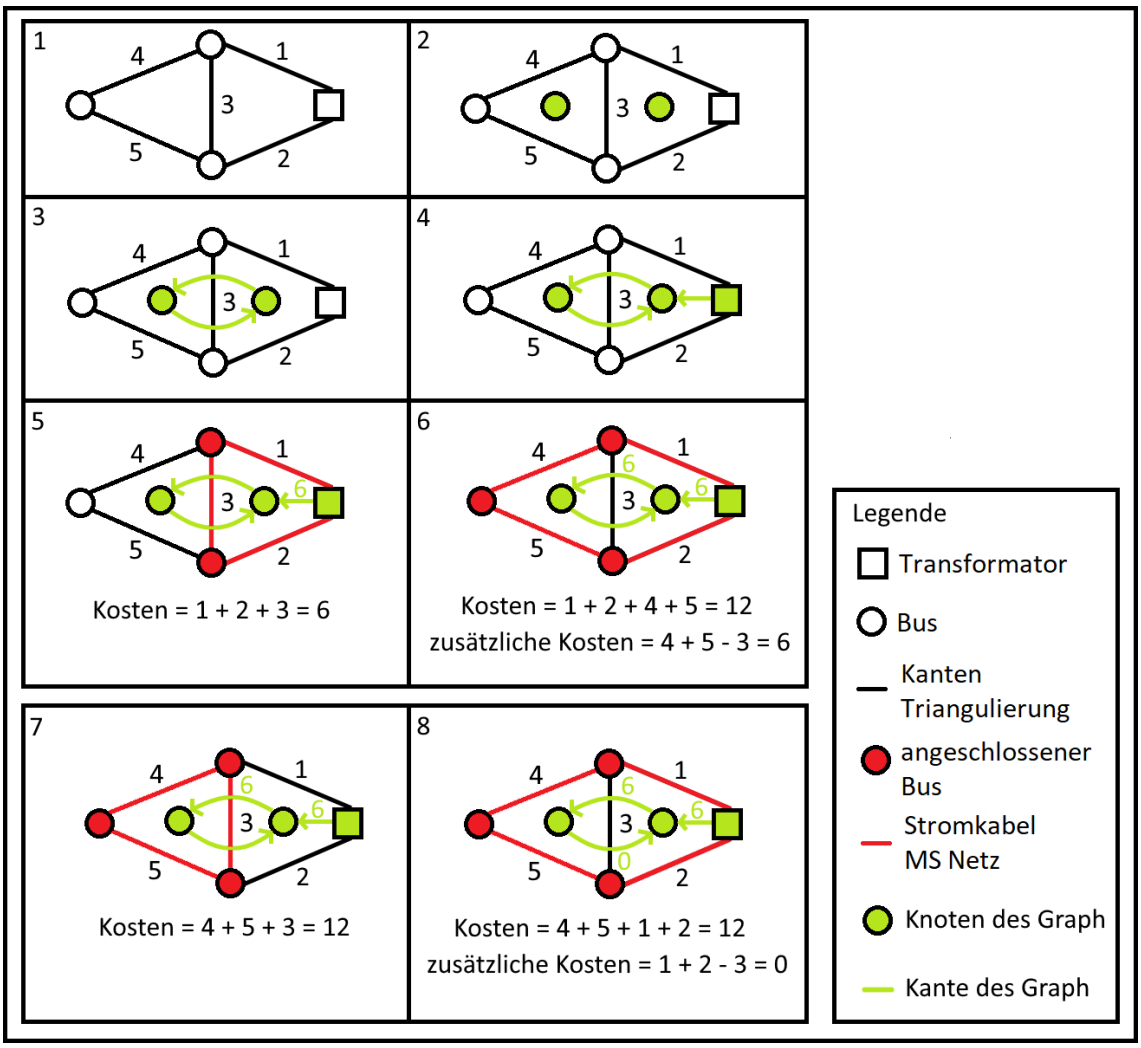


Abbildung 4.1.: Diese Abbildung zeigt die Schritte, welche zur Grapherstellung durchgeführt werden. Zuerst wird jedes Dreieck als Knoten in den neuen Graphen eingefügt. Daraufhin werden die benachbarten Dreiecke miteinander verbunden. Anschließend wird der Transformator als weiterer Knoten in den Graph eingefügt und mit den Knoten der Dreiecke verbunden, in dem er selbst enthalten ist (jedoch nur über ausgehende Kanten). Als nächstes werden die Kantengewichte ermittelt, indem die zusätzlichen Kosten bei der Erweiterung des Mittelspannungsnetzes über diese Kante berechnet werden. In Kachel 7 ist eine Situation zu sehen, welche so nicht möglich ist, da das Stromnetz keinen Transformator enthält. Diese Kachel ist nur dazu da, um zu zeigen, dass die Kosten nicht symmetrisch sind. Quelle: Eigene Darstellung

Abschnitt erwähnt, gibt es keine Kante vom rechten Dreieck zurück zum Transformator, da dieser ja bereits zum Netz gehört, wenn das rechte Dreieck zum Netz gehört. Die sechste Kachel zeigt, wie die Kosten nach der Erweiterung aussehen. Um all diese Leitungen bauen zu können, würden Kosten in Höhe von $1+2+4+5=12$ entstehen. Die zusätzlichen Kosten, die dabei entstehen, sind $4+5-3=6$, da zwei zusätzliche Leitungen mit 4 bzw. 5 Kosten hinzukommen, man jedoch die Leitung mit 3 Kosten einspart. Das Kantengewicht der Kante vom rechten Dreieck zum linken Dreieck beträgt somit 6. An dieser Stelle ist anzumerken, dass die Kantengewichte nicht symmetrisch sind. Das Kantengewicht vom linken Dreieck zum rechten Dreieck beträgt $1+2-3=0$. Die zwei neuen Leitungen kosten also genauso viel, wie die Leitung, welche man durch die Erweiterung einspart. Das Kantengewicht vom linken Dreieck zum rechten Dreieck beträgt somit 0. Dies ist in den Kacheln sieben und acht zu sehen. Hierbei möchte ich allerdings anmerken, dass der Fall in Kachel sieben nur zur Demonstration der Asymmetrie dient, da das Stromnetz aus Kachel 7 so nicht realisiert werden könnte, da dort ein Transformator fehlen würde.

4.5. Optimierungsschleife

Die Optimierungsschleife besteht im Wesentlichen aus der Lösungskonstruktion der Ameisen, womit sich ausführlich in diesem Abschnitt befasst wird. Danach kann optional noch Postprocessing ausgeführt werden, um die Lösung etwas zu verbessern. Diese lokalen Nachbesserungen werden im nächsten Unterkapitel genauer beschrieben. Nach dem Postprocessing wird das lokale Pheromonupdate durchgeführt, um die anderen Ameisen zu weiteren Erkundungen anzuregen. Daraufhin kommen die anderen Ameisen an die Reihe und erstellen ebenfalls ihre Lösungen, führen ggf. ebenfalls Postprocessing durch und wenden ebenfalls das lokale Pheromonupdate an. Nachdem alle Ameisen fertig sind wird von jeder Ameise die Lösung bewertet. Im Anschluss daran, führt die beste Ameise das globale Pheromonupdate durch. Je nach Methode ist mit beste Ameise, die Beste der aktuellen Iteration oder die Beste seit dem Starten der Optimierungsschleife gemeint (siehe Abschnitt 3.3).

In Abbildung 4.2 sieht man den Prozess der Lösungskonstruktion. Die erste Kachel zeigt hierbei die Triangulierung und führt die Namen für die einzelnen Dreiecke ein. In der zweiten Kachel sieht man zudem den Graph, welcher anhand der Triangulierung erstellt wurde. Die Knoten, welche hierbei die jeweiligen Dreiecke repräsentieren, tragen hierbei die Namen, welche in der ersten Kachel eingeführt wurden. Die Kanten und Kantengewichte des Graphen werden zur Übersichtlichkeit in den folgenden Kacheln nicht dargestellt. Lediglich die Kanten, welche die Ameise bereits ausgewählt hat oder in der aktuellen Situation gerade auswählen kann, werden dargestellt. Die bereits ausgewählten Kanten werden in orange dargestellt, die Wählbaren in blau. Die Schritte der ersten beiden Kacheln werden vor der Optimierungsschleife durchgeführt und müssen nicht in jeder Iteration wiederholt werden. In der dritten Kachel beginnt der Aufbau der Lösung, welcher in jeder Iteration

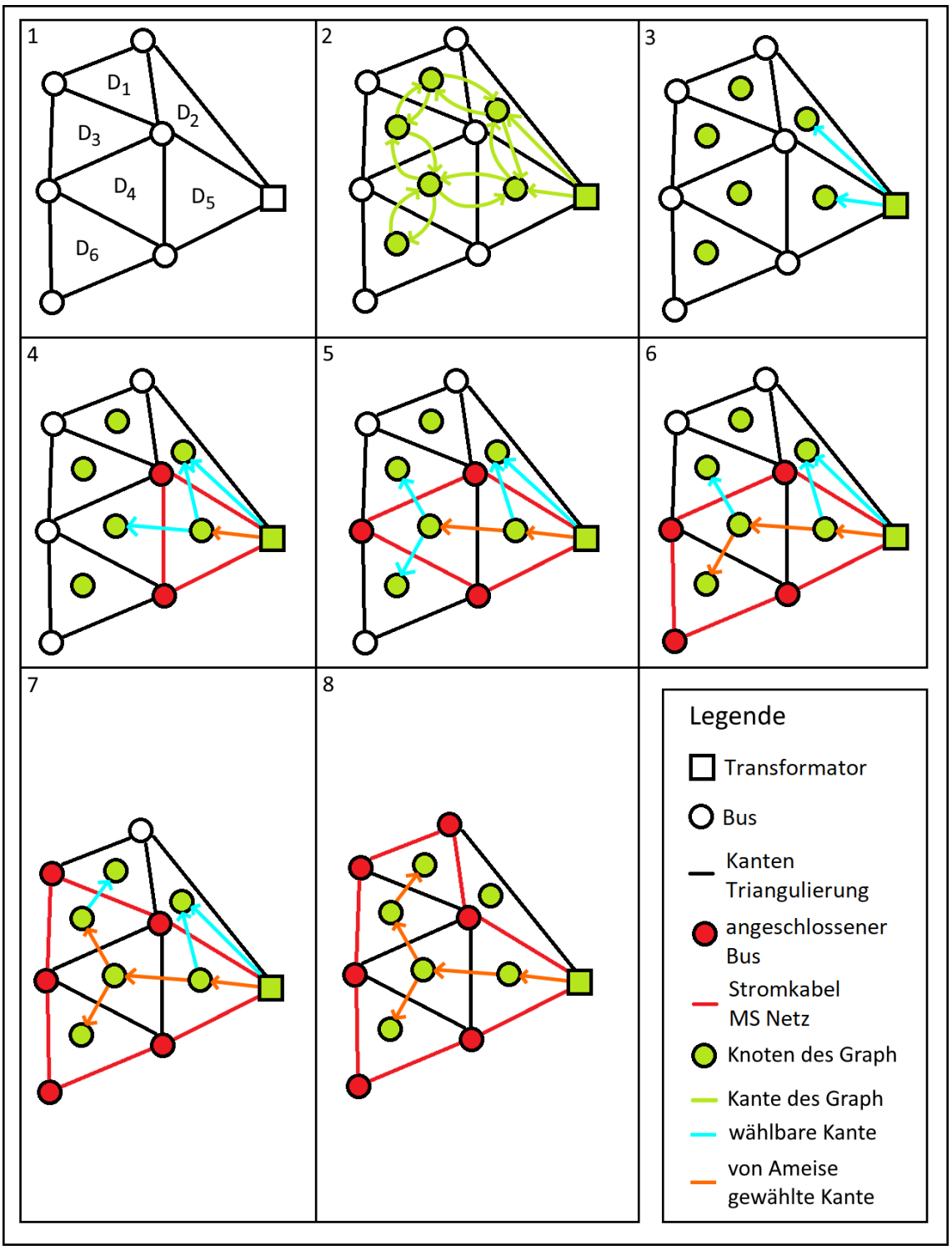


Abbildung 4.2.: Diese Abbildung zeigt die Lösungskonstruktion einer Ameise. Hierbei sieht man in Kachel 1 das Netz, wie es nach der Triangulierung aussieht. In Kachel 2 wird zudem der Graph dargestellt. In den folgenden Kacheln werden von den Kanten nur noch die gewählten bzw. die wählbaren Kanten gezeigt. Nun fügt die Ameise ein Dreieck nach dem anderen hinzu, bis alle Busse verbunden sind. Quelle: Eigene Darstellung

mit einem leeren Mittelspannungsnetz beginnt. Nun hat die Ameise zwei Möglichkeiten. Sie kann das Mittelspannungsnetz mit Dreieck D_5 beginnen, oder die obere blaue Kante wählen und mit D_2 beginnen. In diesem Beispiel wählt die Ameise die untere blaue Kante. Dadurch werden die drei Kanten des Dreiecks D_5 nun an das Mittelspannungsnetz angeschlossen, was in der vierten Kachel zu sehen ist. Das gezeigte Mittelspannungsnetz, bestehend aus dem Transformator, den zwei roten Bussen und den drei Leitungen, ist bereits ein gültiger Ring (vgl. Abschnitt 2.2), auch wenn noch nicht alle Busse angeschlossen sind. Die gewählte Kante wird in orange dargestellt. Von der jetzigen Situation hat die Ameise drei weitere Optionen. Sie kann das Dreieck D_2 vom Dreieck D_5 anschließen und somit den bereits bestehenden Ring erweitern. Sie kann das Dreieck D_2 jedoch auch von dem Transformator aus verbinden und so einen neuen Ring starten. Dieser neue Ring hätte dann die Kante zwischen D_2 und D_5 mit dem anderen Ring gemeinsam. Dies hätte den Vorteil, dass z.B. die Tiefbaukosten nur einmal anfallen würden und in den Graben dann zwei Leitungen kommen. Die Ringe sind dadurch trotzdem komplett voneinander getrennt, die räumliche Nähe kann jedoch ausgenutzt werden. In diesem Beispiel entscheidet sich die Ameise jedoch für die dritte Option, die Erweiterung des Rings von D_5 nach D_4. Die Erweiterung ist in der fünften Kachel zu sehen. Zuerst werden die beiden noch nicht verlegten Leitungen gelegt. Danach wird die Leitung, welche sowohl im Dreieck D_4 als auch im Dreieck D_5 enthalten ist, wieder entfernt, damit das Mittelspannungsnetz weiterhin ein Ring bleibt. Die gewählte Kante (D_5, D_4) wird Orange dargestellt. Die nun verfügbaren Optionen sind erneut in Blau dargestellt. Die Menge W der wählbaren Optionen ist in diesem Fall $\mathrm{W} = \{(D_4, D_3), (D_4, D_6), (D_5, D_2), (T, D_2)\}$. Die Menge W der in diesem Schritt wählbaren Optionen für die Ameise besteht also aus den vier blauen Kanten. T steht für den Knoten des Graphen, welcher den Transformator repräsentiert. Von diesen vier Optionen entscheidet sich die Ameise nun, mithilfe von Gleichung 3.5, für die Kante (D_4, D_6). Das Ergebnis ist in der sechsten Kachel zu sehen. Von dort an erweitert die Ameise das Mittelspannungsnetz über die Kante (D_4, D_3), was in der siebten Kachel zu sehen ist. Um auch noch den letzten Bus anzuschließen, wählt die Ameise die Kante (D_3, D_1). Nun ist die Ameise mit der Lösungskonstruktion fertig. Hierzu möchte ich noch einige Anmerkungen machen. Das Dreieck D_2 war nie Teil des Stromnetzes und es ist auch nicht nötig, alle Dreiecke auszuwählen. Das Dreieck D_4 ist hingegen Teil der Lösung. Durch die Erweiterungen des Netzes wurden jedoch alle Leitungen des Dreiecks wieder entfernt, was jedoch ebenfalls kein Problem darstellt.

Wenn der Parameter postprocessing_in_loop wahr ist, so finden nun auf der gerade erzeugten Lösung einige lokale Nachbesserungen statt. Eine von diesen ist z.B. die Stichoptimierung (siehe Abschnitt 4.6). Im Anschluss darauf findet das lokale Pheromonupdate statt, bevor dann die anderen Ameisen denselben Prozess durchlaufen. Nachdem alle Ameisen fertig sind, findet das globale Pheromonupdate statt. Daraufhin kann auch schon die neue Iteration beginnen, in welcher jede Ameise wieder vor einem leeren Mittelspanungsnetz steht, wie es in der dritten Kachel zu

sehen ist. Bei der erneuten Lösungskonstruktion haben die Ameisen jedoch ein bisschen mehr Wissen als zuvor, durch die zusätzlichen Informationen des vorherigen Pheromonupdates.

Abbildung 4.3 zeigt alle Fälle, welche bei der Lösungskonstruktion auftreten können. Der erste Fall, welcher in den Kacheln 1 und 2 dargestellt ist, stammt vom Niklas Rotering [6]. Die restlichen Fälle wurden von mir hinzugefügt, damit der Algorithmus in mehr Situationen besser reagieren kann.

Die ersten beiden Kacheln zeigen den Fall, welcher am häufigsten auftritt. In diesem gehört eine Kante des neu ausgewählten Dreiecks bereits zum Mittelspannungsnetz. Die Erweiterung findet hierbei so statt, dass die beiden Seiten des neuen Dreiecks ebenfalls zum Mittelspannungsnetz hinzugefügt werden und die gemeinsame Seite wieder vom Netz entfernt wird. Dadurch bleibt das Netz weiterhin ein Ring.

Die Kacheln 3 bis 4 zeigen den Fall, dass bereits zwei Kanten des neuen Dreiecks zum Mittelspannungsnetz gehören und zwar zum selben Ring. Hierzu werden in Kachel 3 zunächst die Namen für die Dreiecke bzw. die Knoten, welche diese Dreiecke repräsentieren, eingeführt. In der vierten Kachel sieht man, dass das Dreieck D_3 zum Netz hinzugefügt werden muss, da man ansonsten nicht das Dreieck D_5 hinzufügen kann. Da Dreieck D_5 noch einen nicht angeschlossenen Bus enthält und dieser auf keinem anderen Weg erreicht werden kann, muss das Dreieck D_3 also hinzugefügt werden. Dies ist jedoch nicht mit der Standardmethode (siehe in den ersten beiden Kacheln) machbar. Deshalb wird hierbei ein neuer Fall eingeführt. So werden keine der Seiten von D_3 entfernt und die letzte Seite, welche bisher noch nicht zum Netz gehört, wird mit einem Hin- und Rückweg ausgebaut. In der Kachel wurden die Verbindungen zu den Bussen nicht ganz geschlossen, um zu verdeutlichen, das diese Leitungen nicht elektrisch mit den Bussen verbunden sind. Dadurch, dass die elektrischen Kontakte nicht bestehen, handelt es sich bei dem Netz weiterhin um einen Ring. Von hier aus kann die Expansion wieder mehr oder weniger normal weitergehen, wie in Kachel 6 gezeigt. Um D_5 zu verbinden, werden die noch nicht zum Mittelspannungsnetz gehörenden Seiten mit dem Mittelspannungsnetz verbunden und auf der anderen Seite wird eine der doppelten Leitungen entfernt. Auf diese Art und Weise wird ebenfalls garantiert, dass es sich um einen Ring handelt. Auch in der sechsten Kachel wurden die Verbindungen zu den Bussen nicht ganz geschlossen, um zu verdeutlichen, dass kein elektrischer Kontakt besteht. In der Realität könnte man dies natürlich durch einen Schalter (siehe Abschnitt 2.1) realisieren und kann die Leitungen dann über einen offenen Schalter mit dem Bus verbinden. Auf diese Art und Weise können z.B. bereits bestehende Leitungen auch für solch eine Leitung verwendet werden. In den Kacheln 7 bis 9 wird dieselbe Expansion gezeigt, diese Mal gehören die Leitungen von D_3 jedoch zu unterschiedlichen Ringen. Zur besseren Übersichtlichkeit wurden die Ringe in unterschiedlichen Farben dargestellt. Hierbei kann man bei der Expansion wie im normalen Fall (siehe in den ersten beiden Kacheln) vorgehen. Das Einzige, was es dabei zu beachten gibt, ist, dass auf einer Seite von D_3 dann zwei Leitungen von unterschiedlichen Ringen sind. Dies ist jedoch nicht schlimm und spart im besten Fall sogar Tiefbaukosten, wenn derselbe

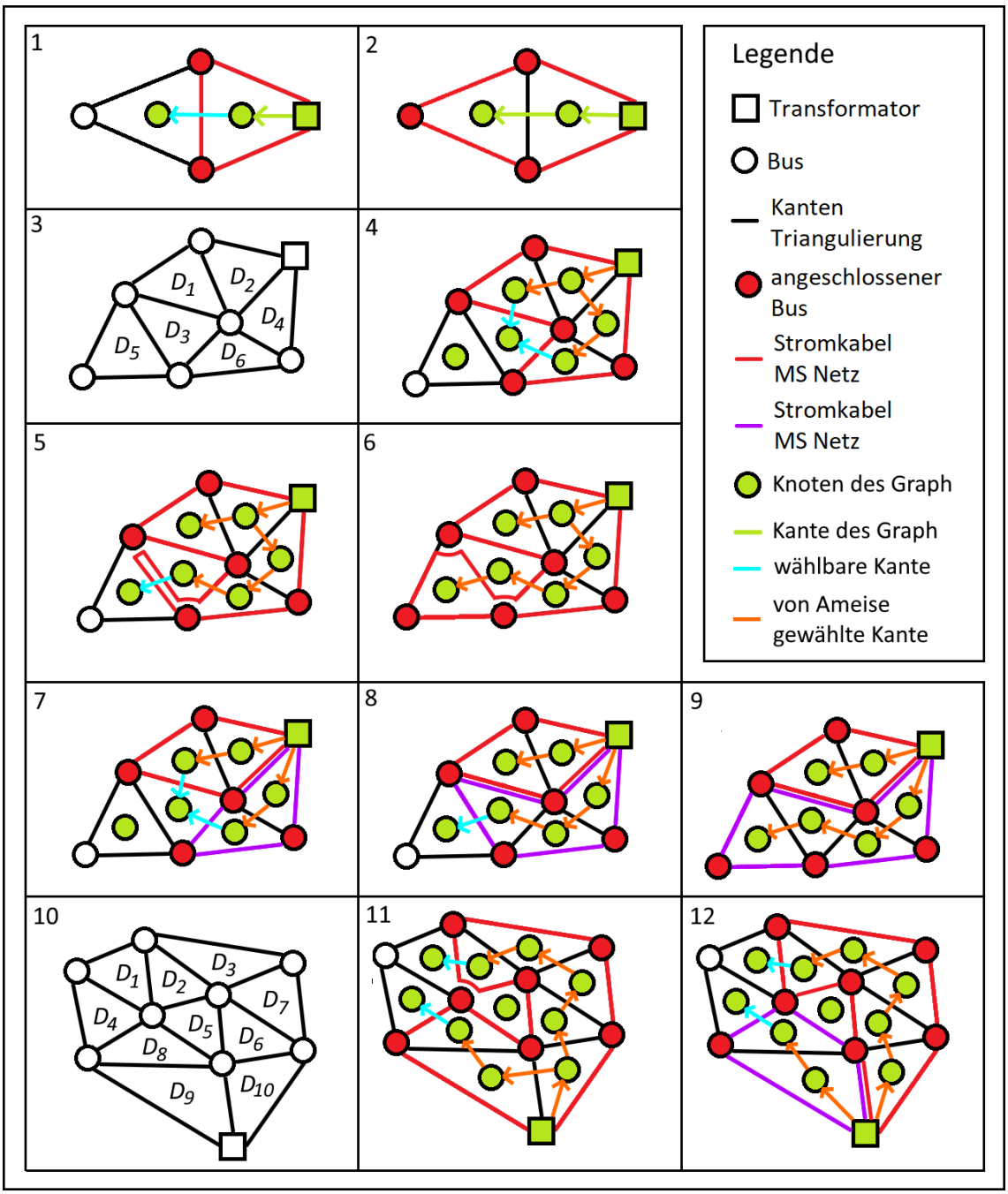


Abbildung 4.3.: Diese Abbildung zeigt alle Fälle, welche bei der Lösungskonstruktion auftreten können. Die ersten beiden Kacheln zeigen den häufigsten Fall, in welchem zwei Leitungen hinzugefügt werden und die gemeinsame Leitung entfernt wird. Kacheln 3 bis 6 zeigen den Fall, in welchem bereits zwei Leitungen von dem Dreieck zum selben Ring gehören. In den Kacheln 7 bis 10 sieht man den Fall, dass bereits zwei Leitungen zum Mittelspannungsnetz gehören, jedoch zu unterschiedlichen Ringen. Die letzten drei Kacheln zeigen den Fall, dass bereits drei Leitungen zum Netz gehören. In diesem Fall kann das entsprechende Dreieck (D_5) nicht mehr ausgewählt werden. Quelle: Eigene Darstellung

Graben für beide Leitungen verwendet werden kann. Nachdem D_3 angeschlossen wurde, kann von dort D_5 verbunden werden, um den letzten Bus anzuschließen, was in der neunten Kachel zu sehen ist.

Die letzten drei Kacheln zeigen den Fall, in welchem bereits alle drei Seiten des Dreiecks zum Mittelspannungsnetz gehören, aber das Dreieck selbst nicht. In diesem Fall handelt es sich um das Dreieck D_5. In Kachel 11 sieht man, dass von diesem bereits alle Seiten Teil des Mittelspannungsnetzes sind, aber D_5 nicht. Wie man jedoch auch sieht, gibt es keine blauen Pfeile zu D_5. Dies liegt daran, dass das Dreieck nicht zum Netz hinzugefügt werden kann, da bereits alle Seiten Teil des Netzes sind. An dieser Situation ändert sich auch nichts, wenn die Seiten Teil von unterschiedlichen Ringen sind, wie es in der zwölften Kachel zu sehen ist.

4.6. Lokale Nachbesserungen

In diesem Abschnitt werden zwei lokale Nachbesserungen vorgestellt. Diese können entweder in jeder Iteration stattfinden oder ein Mal, nachdem alle Iterationen abgeschlossen sind. Hierbei muss man zwischen Laufzeit und Qualität der Lösungen abwägen. Im folgenden wird einerseits die Stichoptimierung vorgestellt, welche Niklas Rotering [6] bereits vorgestellt hat und andererseits die von mir erstellte Strangoptimierung.

4.6.1. Stichoptimierung

Die Stichoptimierung wird für jede Leitung des Mittelspannungsnetzes der erstellten Lösung durchgeführt. Abbildung 4.4 zeigt die Durchführung der Stichoptimierung für die gestrichelte Kante. In der ersten Kachel sieht man die Triangulierung mit den jeweiligen Kantengewichten. Wie man sieht, gibt es vom Transformator aus in diesem Fall nur einen Weg das Netz aufzubauen. Das Ergebnis der Optimierungsschleife ist in der zweiten Kachel zu sehen. Nun beginnt die Stichoptimierung, welche für jede Kante durchgeführt wird. In der Abbildung ist der Fall für die gestrichelte Kante gezeigt. Hierbei bleibt die Kante zunächst mit dem Knoten K_1 verbunden. Statt dem Knoten K_2 wird die Kante nun jedoch mit dessen Nachbarknoten verbunden, was in der dritten Kachel zu sehen ist. Durch diese Änderung könnten die Kosten um 2 reduziert werden, da eine Kante mit Kosten von 5 durch eine Kante mit 3 ersetzt wurde. Wenn man die Kante nun jedoch erneut mit dem Nachbarknoten verbindet, kann eine noch größere Optimierung entstehen. Dieser Fall ist in Kachel 4 zu sehen, in welchem die Kosten um 4 im Vergleich zur Ausgangssituation gesenkt werden konnten. Da ein Stich die maximale Größe von 3 haben darf, wird jetzt erneut noch einmal der Nachbarknoten verbunden. Dieser Fall ist hier jedoch nicht abgebildet. Nun muss zudem noch der symmetrische Fall durchgeführt werden, also dass die gestrichelte Kante mit K_2 verbunden bleibt und K_1 von Nachbarknoten zu Nachbarknoten wechselt. Da die gestrichelte Kante nur eine der Positionen

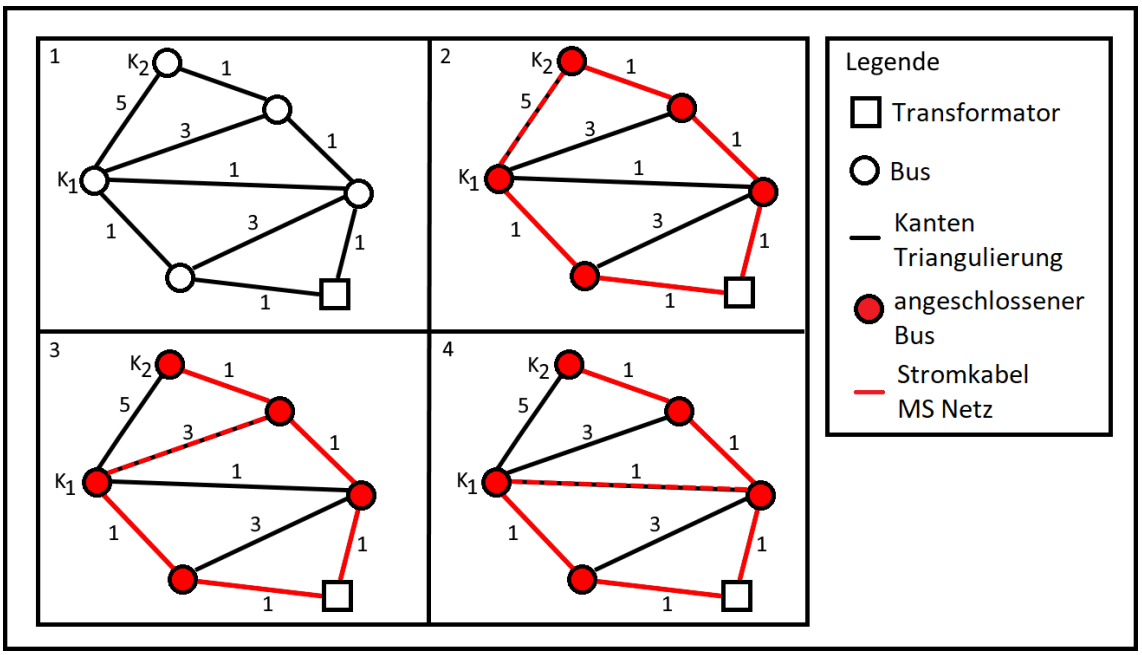


Abbildung 4.4.: Diese Abbildung zeigt die Funktionsweise der Stichoptimierung anhand einer Kante. In der zweiten Kachel sieht man das Netz, wie es nach der Optimierungsschleife aussieht. Für die gestrichelte Kante wird nun geprüft, ob die Stichoptimierung sinnvoll ist. Dies geschieht in der dritten und vierten Kachel. Wie man sieht, lassen sich durch die Stichoptimierung die Kosten reduzieren. Der günstigste Fall ist hierbei in der vierten Kachel zu sehen. Quelle: Eigene Darstellung

annehmen kann, muss man sich merken, welche Möglichkeit die beste Einsparung gebracht hat. Das reicht jedoch noch nicht, da z.B. eine Nachbarkante eine weit größere Einsparung bringen könnte, wenn sie nicht aufgrund der jetzigen Kante bereits in einem Stich wäre. In meiner Implementierung habe ich mich deshalb dafür entschieden, die Fälle nicht direkt zu realisieren, sondern die Optionen in eine Priority Queue einzufügen. Dann wird, wenn von allen Kanten die Optionen eingefügt wurden, die beste Option realisiert. Nun wird nach und nach für jedes Element aus der Priority Queue geprüft, ob die Realisierung noch möglich ist. Daraufhin wird diese, sofern möglich, ausgeführt.

Zudem möchte ich noch anmerken, dass die Stichoptimierung nicht dadurch entsteht, dass man weniger Leitungen verlegen muss, sondern dadurch, dass man eine teure Leitung gegen eine billigere austauscht. In dem Beispiel in Abbildung 4.4 haben alle Mittelspannungsnetze genau 6 Kanten, unabhängig davon, ob es sich um einen reinen Ring oder um einen Ring mit Stich handelt.

4.6.2. Strangoptimierung

Wenn der Algorithmus einen Ring baut, darf er hierbei auch Dreiecke wählen, welche einen Bus mit Transformator enthalten. Dann wird dabei ein Strang (siehe Abschnitt 2.2) erzeugt. In diesem Fall werden sogar stets zwei Stränge erzeugt, da der Ring am Transformator in zwei Stränge geteilt wird. An dieser Stelle greift meine Optimierung, da es manchmal möglich ist, einen der beiden Stränge einzusparen. Abbildung 4.5 zeigt die Situationen, in denen dies möglich ist. In der ersten Kachel sieht man die Ausgangssituation, welche von einer Ameise erzeugt wurde. Man sieht hierbei, dass der rote Ring durch hinzufügen des unteren Transformators in zwei Stränge aufgeteilt wurde, ebenso der lila Ring. Die zwei mittleren Busse können nun über zwei Stränge angeschlossen werden. Die Optimierung versucht nun zu prüfen, ob es möglich ist, alle Busse eines Strangs über einen anderen Strang oder Ring anzuschließen. In der zweiten Kachel ist dies genauer zu sehen. Hierbei versucht die Optimierung zu prüfen, ob die Busse, welche am roten Strang angeschlossen sind, auch anderweitig versorgt werden können. Wie man in der Kachel sieht ist dies auch möglich. Dadurch kann der komplette rote Strang eingespart werden, was in der dritten Kachel zu sehen ist. Die beiden unteren Kacheln zeigen, dass dies nicht nur möglich ist, wenn zwei Stränge nebeneinander sind, es ist genauso möglich, einen Strang einzusparen, wenn dieser neben einem Ring ist. Wie man in diesem Fall sieht, ist es nicht notwendig, für jede Leitung einen Ersatz zu finden, es reicht, wenn alle Busse anderweitig angeschlossen werden können.

4.7. Der gesamte Algorithmus

Algorithmus 2 zeigt den gesamten Algorithmus (Abschnitt 5.2 erläutert die Laufzeiten, welche hier als Kommentare vermerkt sind). Zunächst werden die Eingangsdaten verarbeitet und in eine Form gebracht, mit der die Triangulierung berechnet werden kann. Diese wird im Anschluss daran auch direkt berechnet. Danach wird der Graph, mit Knoten, Kanten und Kantengewichten, erstellt. Nun sind alle Daten für den Ameisenalgorithmus bereit und eine Kolonie kann erstellt werden. Der letzte Schritt ist die Suche der Kolonie. Diese ist in Algorithmus 3 genauer dargestellt. In der inneren Schleife wird hierbei zuerst von einer Ameise eine Lösung erzeugt. Auf dieser Lösung wird anschließend die Strang- und danach die Stichoptimierung durchgeführt. Danach führt die Ameise das lokale Pheromonupdate durch und fügt die erzeugte Lösung zu einer Liste hinzu. Dieser Vorgang wird nun ebenfalls von den anderen Ameisen der Kolonie durchgeführt. Wenn alle Ameisen fertig sind, wird die Statistik geupdatet und das globale Pheromonupdate findet statt. Danach startet das ganze wieder von vorne, indem die List der in dieser Iteration erzeugten Lösungen wieder zurückgesetzt wird. Nach $nr\_of\_itertions$ Durchgängen ist der Algorithmus fertig und die beste gefundene Lösung kann in der Statistik gefunden werden.

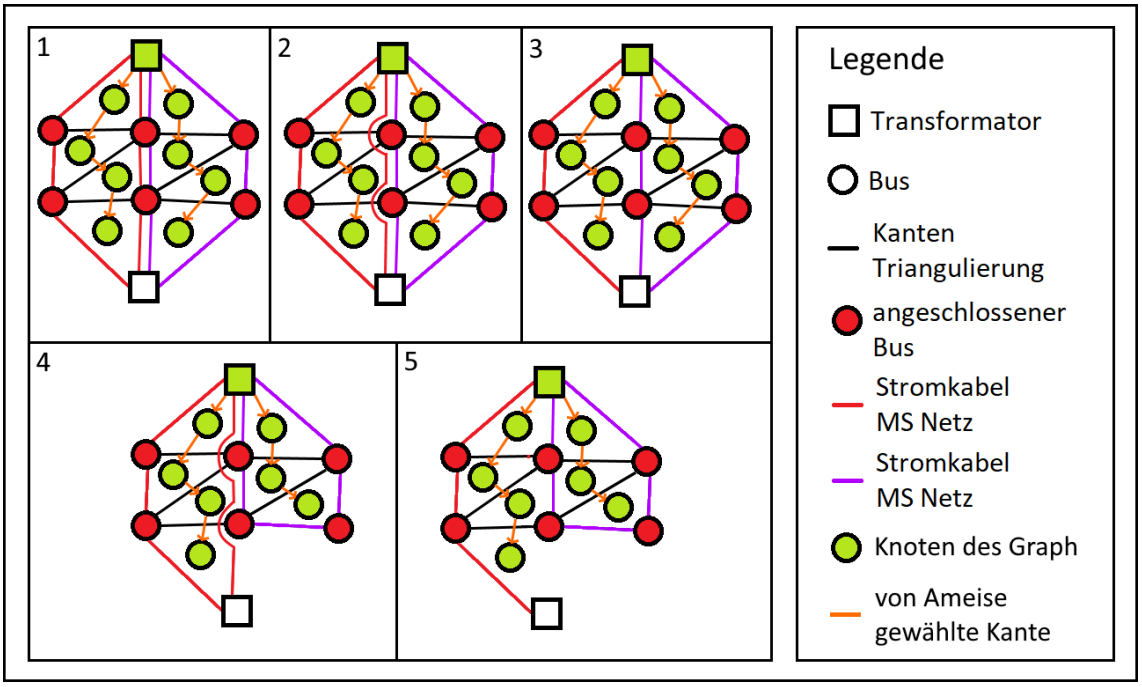


Abbildung 4.5.: Diese Abbildung zeigt die Strangoptimierung. Hierbei wird für jeden Strang geprüft, ob es möglich ist, alle Busse mit über einen anderen Strang oder Ring anzubinden. Wenn dies möglich ist, kann der gesamte Strang eingespart werden. Quelle: Eigene Darstellung

Algorithm 2 Netzplanung

```
data = prepare_triangulation(input_data)                    ▷ O(B)
tri = delaunay_triangulation(data)                          ▷ O(B log B)
tri_and_graph = create_data_structures(tri)                 ▷ O(K²) oder O(K + L)
col = Colony(tri_and_graph)                                 ▷ O(A · K²) oder O(A · (K + L))
col.search()                                                ▷ O(I · A · (B · L²))
```

Algorithm 3 Colony.search()

```
for iteration = 0 to nr_of_iterations − 1 do                ▷ O(I · A · (B · L²))
    solutions = []
    for ant_nr = 0 to nr_of_ants − 1 do                     ▷ O(A · (B · L²))
        solution = ants[ant_nr].construct_solution()        ▷ O(B · L)
        solution = strang_optimization(solution)            ▷ O(B + T² + L²)
        solution = stich_optimization(solution)             ▷ O(L² · B + L log L)
        local_pheromone_update(solution)                    ▷ O(L)
        solutions.add(solution)
    end for
    update_statistics(solutions)                            ▷ O(A)
    global_pheromone_update(solutions)                      ▷ O(A · L)
end for
```

5. Theoretische Analyse

In diesem Kapitel wird der Algorithmus auf theoretischer Ebene analysiert. Hierzu gehören eine Laufzeit- sowie eine Speicherplatzanalyse. Zudem wird die Bedeutung der Parameter beispielhaft am Parameter α erklärt. Das Kapitel wird damit abgeschlossen, dass einige wichtige Eigenschaften bewiesen werden.

5.1. Speicherplatzanalyse

Bevor die Optimierung überhaupt starten kann wird ein Triangulation Objekt benötigt. Dieses enthält einen Graphen, welcher alle B Busse enthält. Wenn der Graph in Matrixform gespeichert wird ist der Speicherbedarf ein Element der Menge $\mathrm{O}(B^2)$, wenn er mithilfe von Adjazenzlisten gespeichert wird, ist der Speicherplatzbedarf hingegen Element der Menge $\mathrm{O}(B + L)$, wobei L die Anzahl der Kanten ist, welche in diesem Fall Leitungen repräsentieren. Aus Implementierungsgründen speichert das Triangulationobjekt ebenfalls einen weiteren Graph, der das Mittelspannungsnetz repräsentiert. Da dieser Graph in der selben Art gespeichert wird, ändert sich an der asymptotischen Laufzeit nichts, da $\mathrm{O}(f(B)) = \mathrm{O}(2 \cdot f(B))$ gilt. Nachdem dieses Triangulationobjekt erzeugt wurde kann eine Kolonie erstellt werden, welcher man das Triangulationobjekt übergibt. Die Kolonie speichert eine Kopie und erzeugt eine voreingestellte Anzahl A an Ameisen. Jede dieser Ameisen erhält eine Kopie des aktuellen Triangulationobjekts, auf welchem die Ameise Veränderungen vornimmt. Insgesamt ist der Speicherbedarf also Element der Menge $\mathrm{O}(A \cdot B^2)$, oder $\mathrm{O}(A \cdot (B + L))$, je nach Implementierung des Graphen.

5.2. Laufzeitanalyse

Algorithmus 2 zeigt den gesamten Algorithmus. Zuerst werden die Daten für die Triangulierung vorbereitet. Die Laufzeit dieses Schritts ist Element der Menge $\mathrm{O}(B)$, da für jeden Bus B eine konstante Anzahl an Änderungen vorgenommen werden muss. Anschließend wird die Triangulierung durchgeführt. Die Laufzeit dafür ist in der Menge $\mathrm{O}(B \log B)$ enthalten. Nach der Triangulierung kann der Graph erstellt werden. Dies beinhaltet das Erstellen der K Knoten, der L Kanten und der Kantengewichte. Je nach Implementierung des Graphen ist die Laufzeit für diesen Schritt in der Menge $\mathrm{O}(K+L)$ oder $\mathrm{O}(K^2)$. Danach kann die Kolonie erstellt werden. Um eine

Kolonie mit A Ameisen zu erstellen, benötigt man A Kopien des Graphen. Je nach Implementierung des Graphen ist dieser Schritt also Element der Menge $O(A \cdot K^2)$ oder $O(A \cdot (K + L))$. Die Laufzeit der anschließenden Optimierungsschleife liegt in $O(I \cdot A \cdot (B \cdot L^2))$. Hierbei ist I die Anzahl der Iterationen. Algorithmus 3 zeigt die Optimierungsschleife im Detail. In der innersten Schleife wird hierbei zuerst von einer Ameise eine Lösung erzeugt. Die Laufzeit dieses Schrittes liegt in $O(B \cdot L)$, weil für jeden Bus durch alle Leitungen iteriert werden muss. Die Laufzeit der anschließenden Strangoptimierung ist Element der Menge $O(B + T^2 + L^2)$. Mit T ist hierbei ein Bus gemeint, an welchen ein Transformator angeschlossen ist. Der Grund für diese Laufzeit liegt darin, dass zuerst einmal über alle Busse iteriert wird, um die Transformatorbusse zu finden. Anschließend wird für jeden Transformatorbus über jeden anderen Transformatorbus iteriert, um herauszufinden, ob die beiden Transformatorbusse an einem gemeinsamen Ring beteiligt sind (wenn dem so ist, handelt es sich eigentlich nicht um einen Ring, sondern um zwei Stränge). Nachdem dies herausgefunden wurde muss für jeden Strang über jede Kante (bzw. Leitung) iteriert werden. Auf die Strangoptimierung folgt die Stichoptimierung. Die Laufzeit der Stichoptimierung ist in der Menge $O(L^2 \cdot B + L \log L)$ enthalten. Dies liegt daran, dass zuerst für jede Kante ein Breitensuchbaum erstellt werden muss. Zudem können während der gesamten Stichoptimierung insgesamt L Kanten in eine Prioritätswarteschlange eingefügt und wieder herausgenommen werden. Im Anschluss an die Stichoptimierung wird das lokale Pheromonupdate durchgeführt. Die Laufzeit dafür ist Element der Menge $O(L)$. Das liegt daran, dass für jede Kante der Lösung ein neuer Pheromonwert berechnet wird. Die innere Schleife wird hierbei durch die Laufzeit, welche Element der Menge $O(L^2 \cdot B) = O(L^2 \cdot B + L \log L)$ ist, dominiert. Alle anderen Laufzeiten der inneren Schleife sind nämlich ebenfalls in dieser Menge enthalten. Da die innerste Schleife A mal durchgeführt wird liegt die Laufzeit für die Durchführung der inneren Schleife in $O(A \cdot B \cdot L^2)$. In der äußeren Schleife wird nach der Durchführung der inneren Schleife nur noch das globale Pheromonupdate durchgeführt und die Statistik geupdatet. Die Laufzeit dafür ist Element der Menge $O(A \cdot L)$. Das liegt daran, dass für jede Ameise die Kosten der erstellten Lösung berechnet werden müssen. Die Laufzeit für die Berechnung der Kosten liegt in der Menge $O(L)$. Danach erhalten alle Kanten der besten Lösung einen neuen Pheromonwert. Da $O(A \cdot L) \in O(A \cdot (B \cdot L^2))$ gilt, liegt die Laufzeit für einen Durchlauf der äußeren Schleife in der Menge $O(A \cdot (B \cdot L^2))$. Diese Schleife wird I mal ausgeführt, weshalb die Laufzeit für die gesamte Optimierungsschleife ein Element der Menge $O(I \cdot A \cdot (B \cdot L^2))$ ist. Die Laufzeit für den gesamten Algorithmus liegt also in $O(I \cdot A \cdot (B \cdot L^2) + B \log B + A \cdot K^2)$ oder in $O(I \cdot A \cdot (B \cdot L^2) + B \log B + A \cdot (K + L))$, je nach Implementierung des Graphen.

5.3. Pheromonanalyse

In diesem Abschnitt wird der Einfluss des Parameters α in Gleichung 3.4 auf die Lösungskonstruktion analysiert. Zu Analysezwecken wird hierbei der Parameter β auf 0 gesetzt. Zunächst sei die Menge der wählbaren Wege $W = \{w_1, w_2\}$. Zudem sei die Optimierungsschleife noch nicht gestartet. Es gilt also $\tau_1 = \tau_2$ und somit $P(Z = w_1) = P(Z = w_2) = 0,5$. In Abbildung 5.1 ist dies im Schnittpunkt der drei Kurven zu sehen. Unabhängig von α haben beide Wege eine 50 % Chance gewählt zu werden. Sei nun die erste Iteration fertig und nur der Weg w_1 habe Pheromone erhalten. Sei $\tau_{ges} = \tau_1 + \tau_2$ und wegen des Pheromonupdates sei $\tau_1 = 0,6\ \tau_{ges}$. Das heißt τ_2 beträgt somit 40 % von τ_{ges}. Nun hängt die Wahrscheinlichkeit gewählt zu werden deutlich von α ab. Wie man Abbildung 5.1 entnehmen kann beträgt die Wahrscheinlichkeit $P(Z = w_1)$ 60 %, wenn $\alpha = 1$ gilt, 69 %, wenn $\alpha = 2$ gilt und 88 %, wenn $\alpha = 5$ gilt. In diesem Beispiel wäre das die Situation nach der ersten Iteration. Gerade am Anfang werden auch schlechte Komponenten durch Zufall häufig gewählt. Wenn nun bereits durch eine kleine Schwankung die Wahrscheinlichkeit auf fast 90 % angehoben wird führt dies dazu, dass diese Komponente sehr wahrscheinlich wieder gewählt wird. Einerseits ist dies schlecht, da so zufällige Fluktuationen verstärkt werden. Andererseits führt dies dazu, dass gute Komponenten, sehr schnell dauerhaft gewählt werden, sofern sie einmal über die 50 % Hürde kommen. Nun gilt es eine Abwägung zwischen diesen beiden Extremen zu finden. Zu α und weiteren Parametern wird deshalb in Kapitel 6 eine Parameteranalyse durchgeführt.

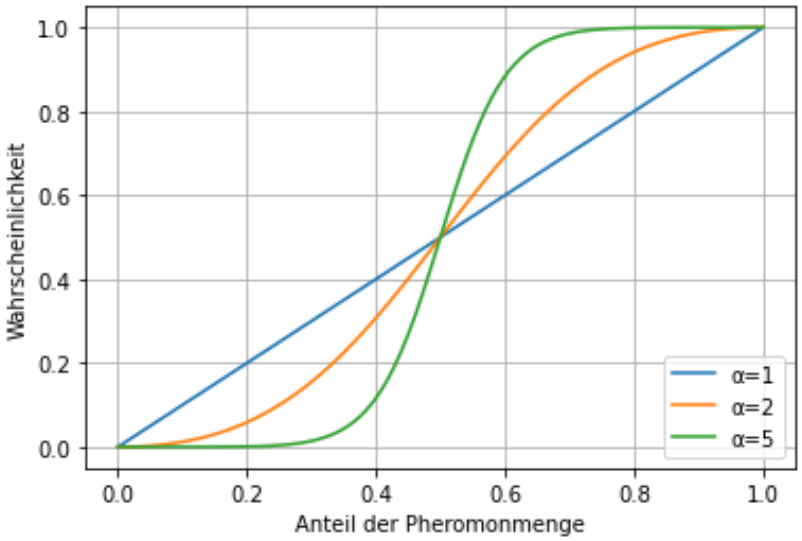


Abbildung 5.1.: Diese Abbildung zeigt, wie groß die Wahrscheinlichkeit einen Weg zu wählen ist, in Abhängigkeit von der prozentualen Anzahl an Pheromonen dieses Weges bezogen auf die Gesamtmenge der aktuell wählbaren Pheromone. Die Wahrscheinlichkeit hängt hierbei zudem vom Parameter α ab. Quelle: Eigene Darstellung

5.4. Beweise einiger Eigenschaften

In diesem Abschnitt geht es darum, einige Aussagen zu beweisen. Die Beweise gehen hierbei sowohl um den Algorithmus, als auch um das Problem der Mittelspannungsnetzplanung. Die Beweise sind einerseits nützlich, da damit Garantien zu den Verfahren gemacht werden können. Andererseits können ebenso Grenzen des Verfahrens aufgezeigt werden. Zudem gibt es noch einige Beweise, welche eher theoretischer Natur sind, da ihre Aussage, z.B. wegen einer zu hohen Laufzeit, keine Relevanz für die Realität bieten.

5.4.1. Satz über eine Untergrenze der Kosten

Der erste Beweis geht um eine Untergrenze der Kosten für ein Mittelspannungsnetz.

Satz 1 (Untergrenze der Kosten für Mittelspannungsnetze)**.** *Eine untere Schranke S ist durch*

$$S = \sum_{i=1}^{nr\_of\_buses} c_{min,\ i}$$

gegeben. Hierbei steht $c_{min,\ 1}$ für die geringsten Kosten, welche eine Kante hat, $c_{min,\ 2}$ für die zweitkleinsten Kosten etc. Hierbei bedeutet zweitkleinste Kosten, dass nach Entfernen der Kante mit den geringsten Kosten keine Kante mit niedrigeren Kosten existiert. Wenn also das niedrigste Kantengewicht 1 ist, dieses jedoch zwei Mal vorkommt, gilt $c_{min,\ 1} = c_{min,\ 2} = 1$.

Beweis. In einem Ring gibt es genauso viele Kanten wie Knoten. Deshalb muss das Mittelspannungsnetz mindestens nr_of_buses Kanten enthalten. Wenn der Ring nun aus den nr_of_buses Kanten mit den kleinsten Kantengewichten besteht, kann man keine Kante austauschen und damit einen billigeren Ring bauen. Somit wurde eine Untergrenze gefunden.

Diese untere Schranke gilt auch, wenn das Mittelspannungsnetz Stiche enthält. Wie in Unterabschnitt 4.6.1 erwähnt wurde, kann ein Stich durch Entfernen einer Kante an einer Stelle und hinzufügen einer Kante an einer anderen Stelle erzeugt werden. Dadurch, dass genauso viele Kanten hinzugefügt wie entfernt werden, bleibt weiterhin bestehen, dass mindestens nr_of_buses Kanten benötigt werden, um ein Mittelspannungsnetz zu bauen. □

Zu diesem Satz sei noch anzumerken, dass es nicht immer möglich ist, ein Mittelspannungsnetz zu bauen, welches auch nur so viel kostet, wie die untere Schranke S.

5.4.2. Satz über die Korrektheit der Lösungskonstruktion

Im Folgenden wird der Beweis darüber erbracht, dass das Verfahren nur gültige Mittelspannungsnetze erzeugt.

Satz 2 (Der Algorithmus konstruiert nur gültige Mittelspannungsnetze)**.** *Die von den Ameisen erzeugten Mittelspannungsnetze erfüllen stets alle Kriterien, welche ein Mittelspannungsnetz erfüllen muss. Dieser Beweis setzt eine gültige Triangulierung voraus, d.h. insbesondere die Busse sind nicht alle auf einer Geraden, sondern so verteilt, dass Dreiecke gebildet wurden.*

Beweis. Induktionsanfang: Zu Beginn besteht das Mittelspannungsnetz nur aus einem Dreieck. Wie bereits in Abschnitt 4.5 beschrieben, erfüllt dieses Dreieck bereits alle Voraussetzungen für ein gültiges Mittelspannungsnetz.

Induktionsschritt: Nun wird gezeigt, dass ein gültiges Mittelspannungsnetz nach der Erweiterung immer noch gültig ist. Für die Expansion gibt es mehrere Möglichkeiten. Alle haben gemeinsam, dass sie nur in Dreiecken stattfinden können, welche noch nicht zum Mittelspannungsnetz dazugehören. Einerseits kann am Transformator ein neuer Ring gestartet werden. Hierbei bleibt das Mittelspannungsnetz gültig, da der neue Ring keinen elektrischen Kontakt zu anderen Ringen hat und ein Ring, welcher nur aus einem Dreieck besteht, ein gültiges Mittelspannungsnetz ist. Andererseits kann die Expansion auch dadurch ausgeführt werden, dass ein bestehender Ring erweitert wird. Hierbei sind alle Fälle in Abschnitt 4.5 genau beschrieben worden (siehe hierzu auch Abbildung 4.3). In jedem Fall bleibt der Zustand des Mittelspannungsnetzes gültig. □

5.4.3. Satz über das Anschließen aller Busse

Im nächsten Satz geht es darum zu beweisen, dass das Verfahren stets eine Lösung konstruiert, bei welcher alle Busse angeschlossen sind.

Satz 3 (Das Verfahren schließt immer alle Busse an)**.** *Während der Lösungskonstruktion werden stets alle Busse so angeschlossen, dass das Stromnetz ein gültiges Mittelspannungsnetz ist. Dieser Beweis setzt eine gültige Triangulierung voraus, d.h. insbesondere die Busse sind nicht alle auf einer Geraden, sondern so verteilt, dass Dreiecke gebildet wurden.*

Beweis. Satz Theorem 2 hat gezeigt, dass das Mittelspannungsnetz nach der Expansion immer noch gültig ist. Nun bleibt jedoch noch zu zeigen, dass durch die Expansion jeder Bus erreicht werden kann.

Hierzu möchte ich zuerst eine Änderung einführen. In Abschnitt 4.5 habe ich gesagt, dass ein Dreieck nicht gewählt werden kann, wenn bereits alle Seiten am Mittelspannungsnetz beteiligt sind, das Dreieck selber aber noch nicht (siehe hierzu

auch Abbildung 4.3). Dies möchte ich für diesen Beweis ändern. Ein solches Dreieck ist nun ebenfalls auswählbar. Wenn es von einer Ameise gewählt wird, wird nichts an dem Mittelspannungsnetz geändert. Es werden also keine Leitungen hinzugefügt oder entfernt. Von diesem Dreieck aus kann keine Expansion stattfinden (da alle benachbarten Dreiecke bereits am Mittelspannungsnetz beteiligt sind und nur unbeteiligte Dreiecke gewählt werden können. Wäre eines der Nachbardreiecke unbeteiligt, wären nicht alle drei Seiten bereits Teil des Mittelspannungsnetzes). Dadurch, dass nichts verändert wird und von dem Dreieck keine Expansion stattfinden kann, sind die beiden Versionen äquivalent im Ergebnis des Mittelspannungsnetzes (lediglich der Graph der ausgewählten Kanten und Dreiecke (bzw. der Knoten welche die Dreiecke repräsentieren) ist anders).

Durch die gerade beschriebene Änderung kann jedes Dreieck ab einem bestimmten Zeitpunkt hinzugefügt werden (sobald mindestens eine Leitung des Dreiecks am Mittelspannungsnetz beteiligt ist oder das Dreieck einen Transformatorbus enthält). In der Optimierungsschleife wird in jedem Schritt ein Dreieck mehr dem Mittelspannungsnetz hinzugefügt. Da es (bei einer endlichen Eingabe) nur eine endliche Anzahl an Dreiecken geben kann, sind nach einer endlichen Anzahl an Schritten alle Dreiecke am Mittelspannungsnetz beteiligt. Da jeder Bus mindestens in einem Dreieck enthalten ist, ist zu diesem Zeitpunkt auch jeder Bus an das Mittelspannungsnetz angeschlossen. □

5.4.4. Satz über die beschränkte Ringgröße

Der folgende Beweis zeigt, dass das Verfahren nicht garantieren kann, dass alle Busse verbunden werden, wenn die maximale Größe des Rings beschränkt ist.

Satz 4. *Wenn die Größe der Ringe auf eine maximale Zahl z beschränkt wird gilt:*

$\forall z \in \mathbb{N}$: *es existiert ein Mittelspannungsnetz, welches nicht realisiert werden kann*

Beweis. Man kann für jedes $z \in \mathbb{N}$ einen Graph konstruieren, für welchen kein Mittelspannungsnetz erstellt werden kann, welches alle Busse verbindet. Der Graph, welcher in Abbildung 5.2 zu sehen ist und beliebig groß werden kann reicht, um das Gegenteil zu beweisen. Der gesamte Graph enthält nur einen Transformator, welcher ganz links ist. Zum Start der Lösungskonstruktion muss der Algorithmus das Dreieck D_1 wählen. Nachdem dieses gewählt wurde, besteht die einzige Expansionsmöglichkeit darin, D_2 als nächstes zu wählen. Generell gilt, wenn der Algorithmus im letzten Schritt D_x gewählt hat, hat er in diesem Schritt nur die Möglichkeit D_{x+1} zu wählen. Um das Dreieck D_{z+1} wählen zu können, müssen also bereits z Dreiecke zum Mittelspannungsnetz, also dem Ring gehören (da der Transformator nur in einem Dreieck enthalten ist, kann kein zweiter Ring gestartet werden). Da die Ringgröße auf z beschränkt wurde, kann D_{z+1} nicht mehr hinzugefügt werden. Somit wurde eine Situation gefunden, in welcher der Algorithmus nicht alle Busse

verbinden kann, da ein Bus des Dreiecks D_{z+1} nicht mit dem Mittelspannungsnetz verbunden ist (in diesem Fall der Bus, der am weitesten rechts ist).

Um bei der Platzierung der Busse stets genügend Platz zu haben, kann man folgendermaßen vorgehen: Man verwendet ein zweidimensionales x-y-Koordinatensystem und platziert den Transformator im Ursprung. Die Busse, welche oberhalb des Transformators sind, müssen auf der Kurve $y = log(x)$ liegen. Die Busse, welche unterhalb des Transformators sind, müssen auf der Kurve $y = -log(x)$ liegen. Da der Logarithmus gegen unendlich strebt, hat man stets genug Platz in y-Richtung. Durch die negative Krümmung der Logarithmusfunktion verhindert man zudem, dass der platzierte Bus noch andere Dreiecke bilden kann, außer den geplanten. □

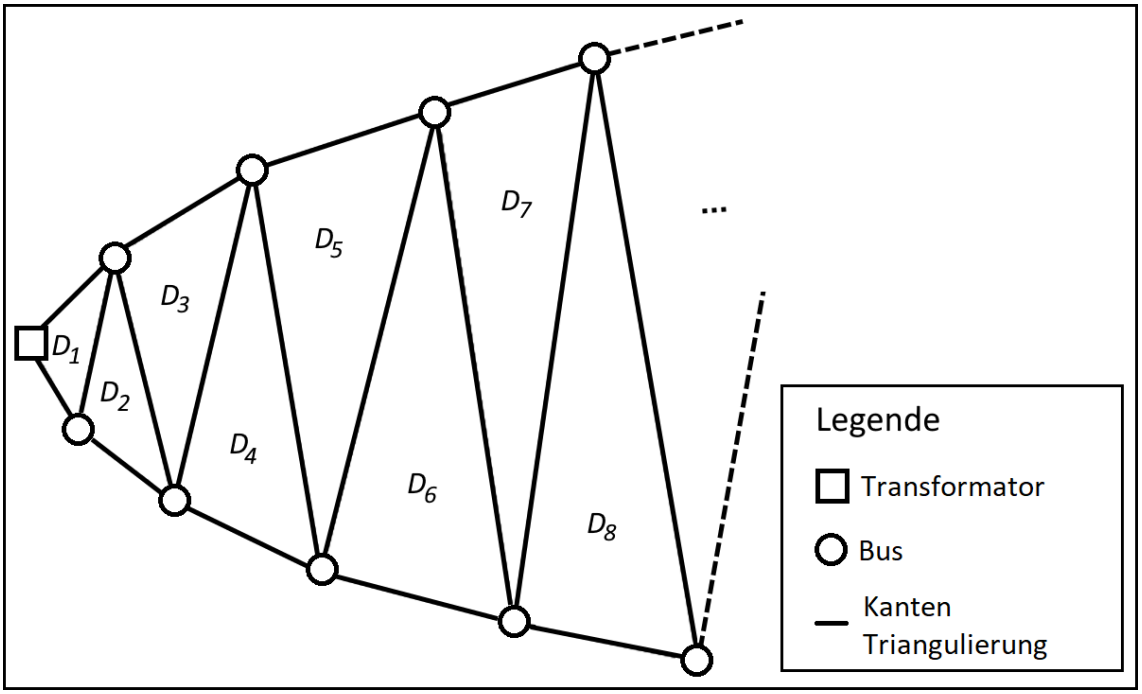


Abbildung 5.2.: Wenn die Busse so angeordnet sind wie auf dieser Abbildung, so kann nach diesem Muster für jede Zahl z eine Situation erstellt werden, in der der Algorithmus einen Ring erzeugen muss, welcher größer als z ist, um alle Busse an den Transformator anzuschließen. Quelle: Eigene Darstellung

5.4.5. Satz über das Finden des globalen Optimums

Im Folgenden möchte ich noch auf eine theoretische Erkenntnis aus dem Buch Ant Colony Optimization [8] eingehen. In diesem wurde bewiesen, dass die Ameisenalgorithmen unter bestimmten Voraussetzungen stets das globale Optimum finden. Diese Voraussetzungen sind gegeben und im Folgenden möchte ich den Beweis dazu vorstellen.

Satz 5. *Der Algorithmus findet immer das globale Optimum, sofern genügend Iterationen durchgeführt werden.*

Beweis. In Abschnitt 3.3 wurde gezeigt, dass die Pheromonkonzentration sowohl nach unten, als auch nach oben beschränkt ist. Nun nimmt man an, dass alle Komponenten der optimalen Lösung die minimale Konzentration haben und alle anderen Komponenten die maximale Pheromonkonzentration haben. Die Wahrscheinlichkeit w, die optimale Lösung zu konstruieren, ist somit gering, aber nicht null. Nach i Iterationen ist die Wahrscheinlichkeit, die optimale Lösung konstruiert zu haben

$$1 - (1 - w)^i.$$

Für $i \to \infty$ geht $(1 - w)^i$ gegen 0 und die Wahrscheinlichkeit, die Lösung bis dahin zu konstruieren, somit gegen 1. □

Der obige Beweis zeigt zwar, dass der Algorithmus selbst im worst case das globale Optimum finden kann. In der Praxis ist dieser Beweis jedoch nicht relevant, da diese Methode der brute force Methode ähnelt. Da die brute force Methode jedoch zu lange dauert, ist dieser Ansatz nicht vielversprechend. Andererseits ist die Idee der Heuristik ja, dass die Komponenten der optimalen Lösung mit den Iterationen immer mehr Pheromone ansammeln und somit die Wahrscheinlichkeit, die optimale Lösung zu konstruieren, deutlich größer ist, als hier im worst case Beispiel.

6. Praktische Analyse

In diesem Kapitel wird der Algorithmus an drei Testnetzen getestet. Die Testnetze wurden so erstellt, dass das globale Optimum der Testnetze bekannt ist und der Algorithmus somit daran bewertet werden kann, ob er das globale Optimum findet, oder wie nah er diesem kommt. Zudem wird der Algorithmus noch mit einem einfacheren Algorithmus verglichen, um zu prüfen, ob die zusätzliche Komplexität, im Vergleich zu einem einfachen Algorithmus, wirklich notwendig ist.

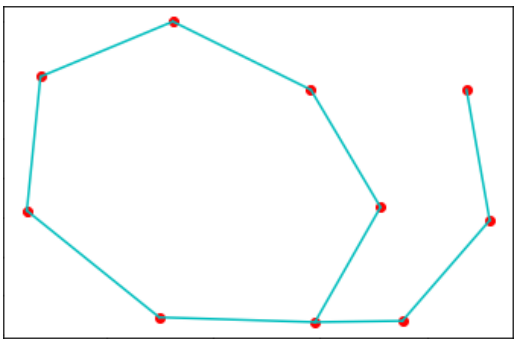
6.1. Die Testnetze

In diesem Abschnitt werden die Testnetze genauer vorgestellt. Zudem wird jeweils bewiesen, wie das globale Optimum aussieht und, dass dieses eindeutig ist. Die ersten beiden Testnetze sind hierbei kleiner und einfacher gehalten. Das dritte Testnetz hingegen enthält mehrere Ringe, Stiche und Stränge und ist zudem deutlich größer.

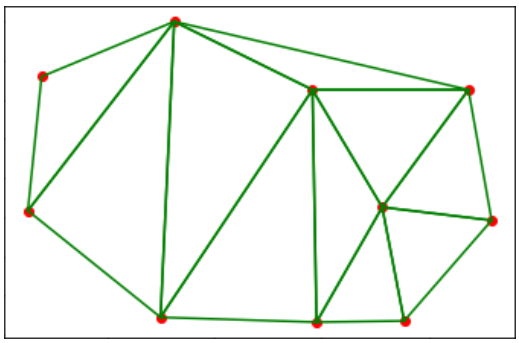
6.1.1. Testnetz mit Stich

Abbildung 6.1 zeigt das erste Testnetz. Hierbei handelt es sich um einen Ring, welche einen Stich der Länge drei hat (siehe Abbildung 6.1a). Die roten Punkte sind hierbei die Busse und die blauen Linien sind die Leitungen des Mittelspannungsnetzes. In Abbildung 6.1b ist die Triangulierung der Busse zu sehen. Der Graph, auf welchem die Ameisen ihre Optimierung durchführen, ist in Abbildung 6.1c zu sehen. Die Triangulierung und der Graph sind zudem in Abbildung 6.1d zu sehen. Hierbei kann man auch erkennen, an welchem Bus der Transformator angeschlossen ist, da dies der einzige Knoten ist, welcher sowohl in der Triangulierung, als auch im Graph enthalten ist.

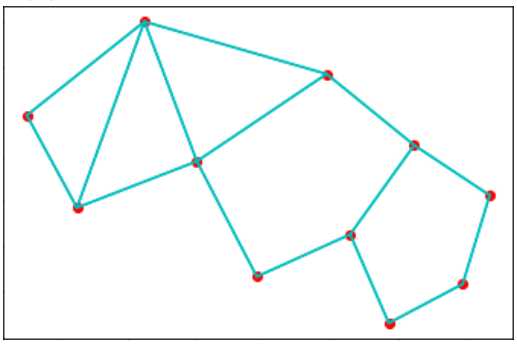
In Abbildung 6.1a sind nur die zehn optimalen Leitungen gezeigt. Jede dieser Leitungen hat ein Kantengewicht von 1. Insgesamt besteht das Testnetz jedoch aus 44 Leitungen. Alle nicht abgebildeten Leitungen haben ein Kantengewicht von 2. Die Verwendung der Triangulierung reduziert die Leitungen von 44 auf 19, wodurch der Algorithmus weniger Leitungen berücksichtigen muss.



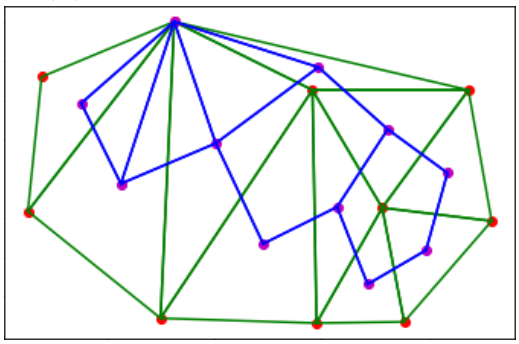
(a) globales Optimum des Testnetzes



(b) Triangulierung des Testnetzes



(c) Graph des Testnetzes



(d) Triangulierung und Graph

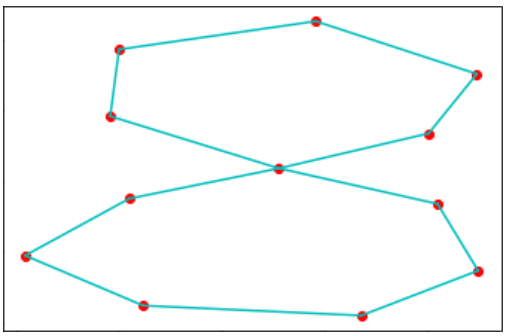
Abbildung 6.1.: In dieser Abbildung ist das erste Testnetz zu sehen. In Abbildung 6.1a sieht man das optimale Mittelspannungsnetz. Abbildung 6.1b zeigt die Triangulierung. Der Graph ist in Abbildung 6.1c zu sehen. Graph und Triangulierung sind in Abbildung 6.1d gemeinsam dargestellt. Quelle: Eigene Darstellung

Satz 6. *Das globale Optimum des ersten Testnetzes, welches in Abbildung 6.1a zu sehen ist, ist eindeutig und liegt bei 10.*

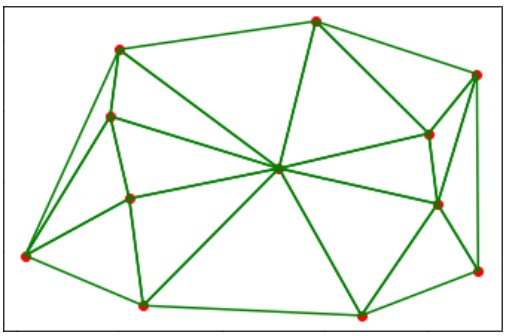
Beweis. Nach Satz Theorem 1 liegt die Untergrenze für dieses Testnetz bei 10. Diese Untergrenze ist auch erreichbar, da genau die zehn Leitungen aus Abbildung 6.1a das Kantengewicht von 1 haben. Dadurch, dass es nur diese zehn Kanten mit minimalen Gewicht gibt und alle anderen Leitungen ein größeres Kantengewicht haben, ist diese Lösung auch eindeutig. □

6.1.2. Testnetz mit zwei Ringen

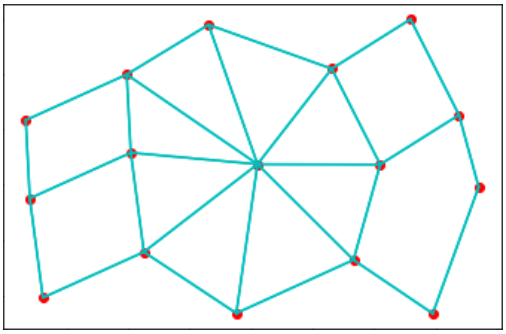
Abbildung 6.2 zeigt das zweite Testnetz. Die optimale Lösung dieses Testnetzes besteht aus zwei Ringen, welche vom Transformator in der Mitte starten. Dieses optimale Netz ist in Abbildung 6.2a zu sehen. Abbildung 6.2b zeigt die Triangulierung und in Abbildung 6.2c ist der Graph zu sehen. Der Graph und die Triangulierung sind zudem in Abbildung 6.2d gleichzeitig dargestellt.



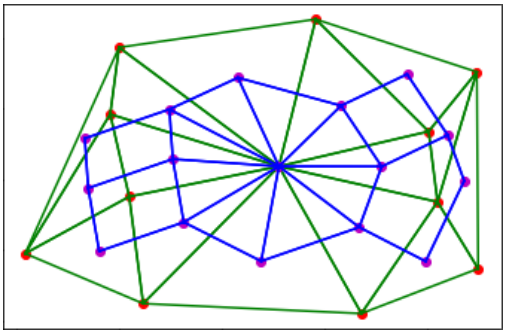
(a) globales Optimum des Testnetzes



(b) Triangulierung des Testnetzes



(c) Graph des Testnetzes



(d) Triangulierung und Graph

Abbildung 6.2.: Diese Abbildung zeigt das zweite Testnetz. In Abbildung 6.2a ist das globale Optimum des Testnetzes zu sehen. Abbildung 6.2b zeigt die Triangulierung und in Abbildung 6.2c ist der Graph zu sehen. Graph und Triangulierung sind in Abbildung 6.2d gemeinsam dargestellt. Quelle: Eigene Darstellung

In Abbildung 6.2a sind nur die 13 optimalen Leitungen gezeigt. Jede dieser Leitungen hat ein Kantengewicht von 1. Das gesamte Testnetz besteht hingegen aus 103 Kanten. Durch die Triangulierung wird die Anzahl der Kanten jedoch auf 26 reduziert. Alle nicht abgebildeten Kanten haben ein Kantengewicht von 2,5.

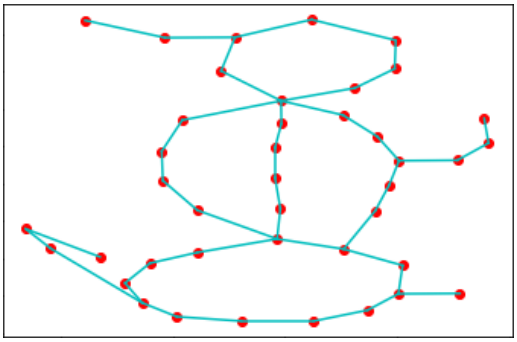
Satz 7. *Das globale Optimum des zweiten Testnetzes, welches in Abbildung 6.2a zu sehen ist, ist eindeutig und liegt bei 13.*

Beweis. Das in Abbildung 6.2a dargestellte Netz ist gültig und die Kosten für dieses Netz liegen bei 13. Nach Satz Theorem 1 liegt die untere Schranke für dieses Testnetz bei 12. Das liegt daran, dass ein Netz mit 12 Bussen mindestens 12 Kanten benötigt und die billigsten 12 Kantengewichte zusammen ein Gewicht von 12 haben. Würde man jedoch eine der 13 Kanten aus Abbildung 6.2a entfernen, würde es sich nicht mehr um ein gültiges Mittelspannungsnetz handeln. Deshalb gibt es keine Lösung, welche aus 12 Kanten besteht, welche alle ein Kantengewicht von 1 haben. Es müsste also mindestens eine der 12 Kanten ein Kantengewicht von 2,5 haben. Dadurch würden die Kosten jedoch mindestens bei $11 \cdot 1 + 2,5 = 13,5$ liegen. Somit ist die Lösung mit 13 Kosten am günstigsten. Da es nur 13 Kanten mit einem Kantengewicht von 1 gibt, ist diese Lösung auch eindeutig. □

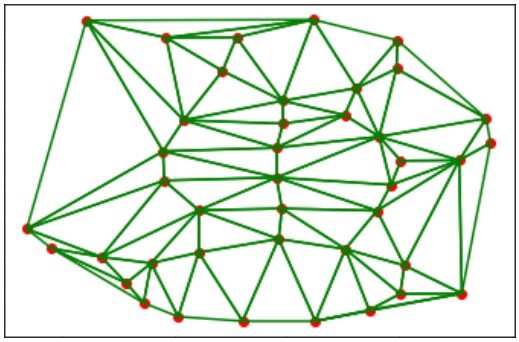
6.1.3. Kombinierter Testfall

In Abbildung 6.3 ist das dritte Testnetz zu sehen. Das globale Optimum ist in Abbildung 6.3a zu sehen. Es besteht aus zwei Ringen, drei Strängen und insgesamt vier Stichen. Die vier Stiche haben eine Länge von eins bis drei und decken somit alle erlaubten Längen ab. Abbildung 6.3b zeigt die Triangulierung und in Abbildung 6.3c ist der Graph zu sehen. Der Graph und die Triangulierung sind zudem nochmals gemeinsam in Abbildung 6.3d dargestellt.

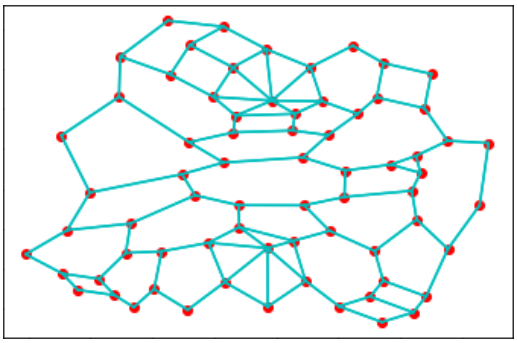
Abbildung 6.3a zeigt nur die 44 Kanten, welche Teil der globalen Optimallösung sind. Jede dieser 44 Kanten hat ein Kantengewicht von 1. Die anderen 1390 Kanten haben ein Kantengewicht von 4,5. Durch die Triangulierung wird die Anzahl der Kanten, welche betrachtet werden, auf 106 heruntergebracht.



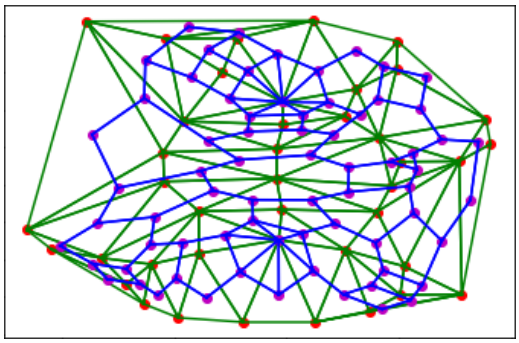
(a) globales Optimum des Testnetzes



(b) Triangulierung des Testnetzes



(c) Graph des Testnetzes



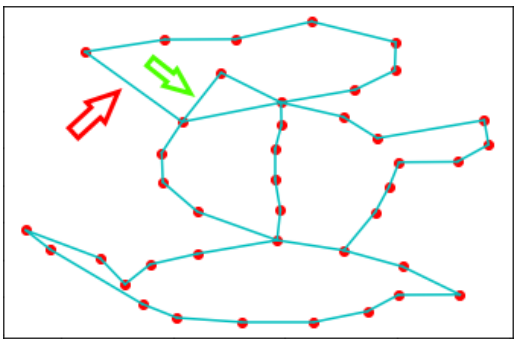
(d) Triangulierung und Graph

Abbildung 6.3.: In dieser Abbildung ist das dritte Testnetz zu sehen. Abbildung 6.3a zeigt das globale Optimum des Testnetzes. Abbildung 6.3b zeigt die Triangulierung und in Abbildung 6.3c ist der Graph zu sehen. In Abbildung 6.3d sind Graph und Triangulierung gemeinsam dargestellt. Quelle: Eigene Darstellung

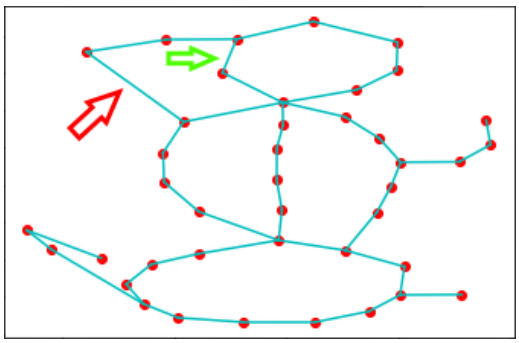
Satz 8. *Das globale Optimum des dritten Testnetzes, welches in Abbildung 6.3a zu sehen ist, ist eindeutig und liegt bei 44*

Beweis. Das in Abbildung 6.3a dargestellte Netz ist gültig und die Kosten für dieses Netz liegen bei 44. Nach Satz Theorem 1 liegt die untere Schranke bei 41, da das Netz 41 Busse hat und somit mindestens 41 Kanten benötigt werden. Die 41 kleinsten Kanten haben alle ein Kantengewicht von 1, was also eine untere Schranke von 41 ergibt. Würde man eine der Kanten aus Abbildung 6.3a entfernen, so würde es sich nicht mehr um ein gültiges Mittelspannungsnetz halten. Auch das Entfernen von weiteren Kanten, würde nicht dazu führen, dass das Netz gültig werden würde. Deshalb kann es keine Lösung geben, welche nur aus Kanten mit einem Kantengewicht von 1 besteht. Für eine Lösung, welche nur aus 41 Kanten besteht, müsste also mindestens eine Kante ein Kantengewicht von 4,5 haben. Dadurch würden die Kosten bei $40 \cdot 1 + 1 \cdot 4,5 = 44,5$ liegen. Somit ist die Lösung, bestehend aus den 44 Kanten mit einem Kantengewicht von 1, das globale Optimum. Da es nur 44 Kanten mit einem Kantengewicht von 1 gibt, ist diese Lösung auch eindeutig. □

An dieser Stelle möchte ich zudem noch ein Beispiel zeigen, in welchem es aufgrund der Triangulierung nicht möglich ist, das globale Optimum zu erreichen. Einerseits kann durch die Triangulierung natürlich eine der Kanten, welche Teil der optimalen Lösung ist, nie betrachtet werden. Das würde passieren, wenn diese Kante nicht Teil der Triangulierung wäre. Aber selbst wenn alle Kanten der optimalen Lösung in der Triangulierung enthalten sind kann es passieren, dass das Optimum nicht erreicht werden kann. Ein solcher Fall ist in Abbildung 6.4 zu sehen. Im Vergleich zu Abbildung 6.3a wurde lediglich der Bus, welcher ganz oben links ist, ein bisschen nach unten verschoben. Die optimalen Kanten sind weiterhin dieselben wie in Abbildung 6.3a. Durch die Verschiebung des Busses wurde die Triangulierung jedoch verändert und nun ist die optimale Lösung nicht mehr erreichbar. Dies liegt daran, dass die beiden markierten Kanten Teil des oberen Rings sein müssen. Durch die Stichoptimierung kann jedoch nur eine der beiden Kanten verschoben werden. Abbildung 6.4b zeigt die Situation nach der Stichoptimierung. Die grün markierte Kante aus Abbildung 6.4a wurde verschoben, wodurch eine Kante, welche nicht Teil der optimalen Lösung ist, entfernt wurde und eine Kante, welche Teil der optimalen Lösung ist, hinzugefügt wurde. Die rot markierte Kante kann jedoch nicht entfernt werden, wodurch sie Teil der Lösung bleibt, obwohl sie nicht zur Optimallösung dazugehört. Das Testnetz aus Abbildung 6.4b enthält also die 44 Kanten mit dem Kantengewicht von 1 und zudem die rot markierte Kante mit dem Kantengewicht von 4,5.



(a) Lösung, welche von einer Ameisen konstruiert werden würde



(b) Lösung, nach der Stich und Strang Optimierung

Abbildung 6.4.: Diese Abbildung zeigt einen Fall, in welchem das globale Optimum nicht erreicht werden kann. In Abbildung 6.4a sieht man die Lösung, wie sie von einer Ameise konstruiert werden würde. In Abbildung 6.4b sieht man dieselbe Lösung nach der Stichoptimierung. Hierbei wird die grüne Kante mithilfe der Stichoptimierung verändert, die rote Kante jedoch nicht. Quelle: Eigene Darstellung

6.2. Vergleichsverfahren

In diesem Abschnitt wird der Algorithmus erklärt, mit welchem der Ameisenalgorithmus verglichen wird. Algorithmus 4 zeigt den Pseudocode des greedy Algorithmus. Der Algorithmus benötigt ein Triangulation Objekt als Eingabe. Dann beginnt er damit, eine Lösung aufzubauen. Solange noch nicht alle Busse verbunden sind, ermittelt er zuerst, welche aktuellen Expansionsmöglichkeiten er hat. Von diesen Verfügbaren wählt er dann die beste Möglichkeit aus und fügt diese der Lösung hinzu. Wenn auf diese Art und Weise alle Busse verbunden wurden, ist der Algorithmus fertig und gibt die konstruierte Lösung zurück. Dieses Verhalten kann mithilfe des Ameisenalgorithmus simuliert werden. Wenn man den Parameter α auf 0 setzt, werden sämtliche Pheromonwerte ignoriert. Zudem muss noch q_0 auf 1 gesetzt werden, damit in jedem Fall die beste Lösung gewählt wird und nicht nur zu $q_0\%$ (siehe hierzu Formel Gleichung 3.5).

An dieser Stelle möchte ich noch bemerken, dass dieser greedy Algorithmus von seiner Art her ähnlich zum Ameisenalgorithmus ist, insbesondere der Ant Colony System Variante. Die Ant Colony System Variante, welche hier verwendet wird, wählt nämlich ebenfalls zu $q_0\%$ die beste verfügbare Option (berücksichtigt dabei aber auch die Pheromone). Nur zu $(1-q_0)$ % wählt der Ameisenalgorithmus eine zufällige Option. Die Algorithmen haben also eine gewisse Ähnlichkeit. Der Vorteil des greedy Algorithmus ist, dass er besonders schnell ist, da er nur eine Lösung erzeugen muss. Die Ameisen hingegen müssen über viele Iteration viele Lösungen erzeugen. Insgesamt $nr\_of\_ants \cdot nr\_of\_iterations$ Lösungen müssen erzeugt werden. Dies benötigt natürlich deutlich mehr Zeit. Dafür sammeln die Ameisen über die Iterationen Wissen, in Form von Pheromonen, an. Um zu prüfen, ob sich diese deutlich

größere Laufzeit lohnt, werden die in Abschnitt 6.1 vorgestellten Testnetze sowohl von dem greedy Algorithmus, als auch von dem Ameisenalgorithmus optimiert und die erzeugten Lösungen miteinander verglichen.

Algorithm 4 Greedy Algorithmus

```
function GREEDY_ALGORITHM(triangulation)
    solution = ∅
    while ¬ triangulation.all_buses_connected(solution) do
        expansion_options = triangulation.get_expansion_options(solution)
        best_component = argmin(expansion_options.cost)
        solution = solution ∪ best_component
    end while
    return solution
end function
```

6.3. Aufbau der Parameter- und Vergleichsstudie

In diesem Abschnitt wird die Vorgehensweise für den Vergleich der Parameter sowie dem Vergleich der Algorithmen erläutert. Tabelle 6.1 zeigt die Standardwerte, welche die Parameter des Ameisenalgorithmus typischerweise haben. Es handelt sich hierbei um Werte, welche aus dem Ant Colony Optimization Buch [8] übernommen wurden, um Erfahrungswerte aus der Masterarbeit von Lukas Gebhard [11] oder um eigene Erfahrung. Mit dieser Studie sollen jedoch die optimalen Parameter für diesen Algorithmus gefunden werden. Im folgenden Experiment wurden dazu alle Werte bis auf einen auf diesem Standardwert gelassen und der eine Wert wurde variiert. Dies ist in Tabelle 6.2 zu sehen. Dem Versuchsnamen kann man entnehmen, welcher Parameter verändert wurde und auf welchen Wert er verändert wurde. Der Versuchsname q095-x bedeutet, dass der Parameter q_0 auf den Wert 0,95 bzw. 95 % gesetzt wurde. Das x wird später durch 1, 2 oder 3 ersetzt und gibt an, ob diese Parameter auf Testnetz 1, 2 oder 3 angewendet wurden.

Tabelle 6.1.: Standardwerte für die Parameter des Ameisenalgorithmus

Parameter	α	β	q_0	ρ	ξ	τ_0
Standardwert	1	1	0,9	0,1	0,1	0,05

Tabelle 6.2.: Aufbau der Parameter- und Vergleichsstudie

Versuchsname	α	β	q_0
greedy	0	1	1
q095-x	1	1	0,95
q09-x	1	1	0,9
q085-x	1	1	0,85
q08-x	1	1	0,8
q075-x	1	1	0,75
a1-x	1	1	0,9
a12-x	1.2	1	0,9
a15-x	1.5	1	0,9
a2-x	2	1	0,9
a3-x	3	1	0,9
b1-x	1	1	0,9
b12-x	1	1.2	0,9
b15-x	1	1.5	0,9
b2-x	1	2	0,9
b3-x	1	3	0,9

Versuchsname	ρ	ξ	τ_0
r005-x	0.05	0.1	0.05
r01-x	0.1	0.1	0.05
r015-x	0.15	0.1	0.05
r02-x	0.2	0.1	0.05
r025-x	0.25	0.1	0.05
x01-x	0.1	0.1	0.05
x005-x	0.1	0.05	0.05
x015-x	0.1	0.15	0.05
x02-x	0.1	0.2	0.05
x025-x	0.1	0.25	0.05
t0005-x	0.1	0.1	0.005
t001-x	0.1	0.1	0.01
t0025-x	0.1	0.1	0.025
t005-x	0.1	0.1	0.05
t01-x	0.1	0.1	0.1

6.4. Ergebnisse und Analyse

In diesem Abschnitt werden die Ergebnisse der Parameterstudie vorgestellt und analysiert.

6.4.1. Testnetz 1

In diesem Abschnitt werden die Ergebnisse der Parameterstudie bezüglich des ersten Testnetzes diskutiert. Die optimale Lösung für das erste Testnetz bestand aus einem Ring, welcher einen Stich der Länge 3 hat. Die Ergebnisse sind in Tabelle 6.3 zu sehen. Da es sich um ein kleineres Testnetz handelt, hatten die Parameter keinen allzu großen Einfluss und mit allen Parametervariationen konnte das Optimum gefunden werden. In Abbildung 6.5 kann man den Einfluss des Parameters α auf die Kosten der gefunden Lösung in Abhängigkeit der Iteration sehen. Wie man sieht wurde bereits in der ersten Iteration die optimale Lösung von 10 gefunden. Diese Lösung wurde auch in jeder Iteration erneut konstruiert, da die minimalen Kosten pro Iteration ebenfalls stets bei 10 liegen. Anhand des Graphen, welcher die durchschnittlichen Kosten pro Iteration zeigt, kann man jedoch erkennen, dass die Ameisen nicht nur die ganze Zeit die optimale Lösung bauen, sondern ebenfalls andere Optionen erkunden. Abbildung 6.5d zeigt die Kosten der Lösung von jeder Ameise. Die Graphen der anderen Parametervariationen sehen denen aus Abbildung 6.5 sehr ähnlich, da das Testnetz recht klein ist. Sämtliche Graphen sind in Anhang A zu finden.

Tabelle 6.3.: Ergebnisse der Parameterstudie für das erste Testnetz

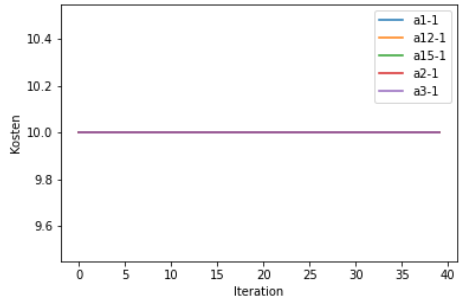
Versuchsname	a1-1	a12-1	a15-1	a2-1	a3-1
globales Optimum gefunden	✓	✓	✓	✓	✓
Versuchsname	b1-1	b12-1	b15-1	b2-1	b3-1
globales Optimum gefunden	✓	✓	✓	✓	✓
Versuchsname	q095-1	q09-1	q085-1	q08-1	q075-1
globales Optimum gefunden	✓	✓	✓	✓	✓
Versuchsname	r01-1	r005-1	r015-1	r02-1	r025-1
globales Optimum gefunden	✓	✓	✓	✓	✓
Versuchsname	x01-1	x005-1	x015-1	x02-1	x025-1
globales Optimum gefunden	✓	✓	✓	✓	✓
Versuchsname	t0005-1	t001-1	t0025-1	t005-1	t01-1
globales Optimum gefunden	✓	✓	✓	✓	✓

6.4.2. Testnetz 2

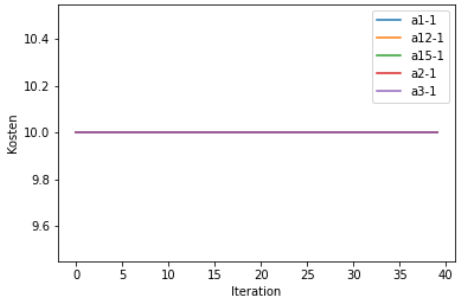
Tabelle 6.4 zeigt die Ergebnisse der Parameterstudie für das zweite Testnetz. Die Optimallösung dieses Testnetzes besteht aus zwei Ringen. Wie man sieht, wurde fast immer das globale Optimum gefunden. Lediglich in zwei Fällen wurde das globale Optimum nicht gefunden. Abbildung 6.6 zeigt die Kostenverläufe in Abhängigkeit des Parameters τ_0. Wie man sieht schaffen es die Ameisen nicht das globale Optimum zu finden, wenn $\tau_0 = 0,01$ beträgt. Ansonsten findet jedoch in fast jeder Iteration mindestens eine Ameise die Optimallösung. Abbildung 6.6c zeigt die durchschnittlichen Kosten pro Iteration. Wie man sieht fällt der Durchschnitt schnell ab, da die Ameisen aufgrund der Pheromone schnell gute Lösungskomponenten bevorzugen. Wie man jedoch anhand von den Kosten pro Ameise pro Iteration erkennen kann erkunden die Ameisen trotzdem noch viele weitere Lösungsmöglichkeiten. Alle Graphen zu jedem Parameter sind in Anhang A zu finden.

6.4.3. Testnetz 3

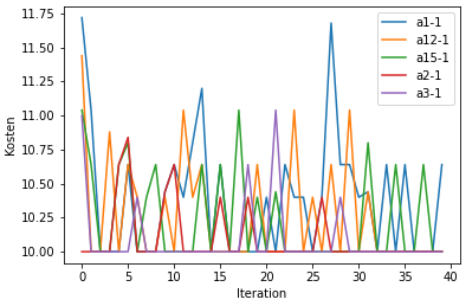
Tabelle 6.5 zeigt die Ergebnisse der Parameterstudie für das dritte Testnetz. Da es keine der Variationen geschafft hat die optimale Lösung zu erstellen wurde stattdessen jeweils die günstigste gefundene Lösung dargestellt. Die beste gefundene Lösung hat Kosten von 60 und liegt damit 16 vom globalen Optimum entfernt, welches bei 44 liegt. Die beste gefundene Lösung ist also 36,4 % größer als die optimale Lösung. Abbildung 6.7 zeigt den Einfluss von ρ auf die Kosten. Anhand der minimalen Kosten seit Beginn kann man sehen, dass die Ameisen bei allen Parameterwerten Wissen ansammeln und anhand von diesem Wissen bessere Lösungen erstellen können. Besonders gut schneiden hierbei die Varianten ab, bei welchen $\rho = 0,1$ und $\rho = 0,15$ gilt. Die minimalen Kosten pro Iteration und auch die durchschnittlichen Kosten pro Iteration zeigen ebenfalls, dass die erstellten Lösungen der Ameisen im



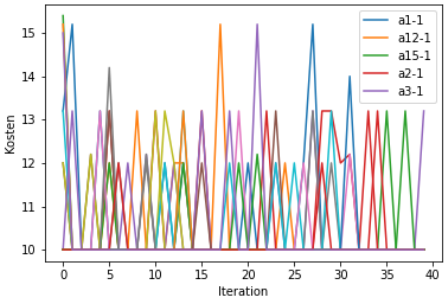
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration



(c) Durchschnittskosten pro Iteration



(d) Kosten pro Ameise pro Iteration

Abbildung 6.5.: Einfluss von α auf die Kosten von Testnetz 1. Quelle: Eigene Darstellung

Verlauf der Iterationen meist besser werden. Abbildung 6.7d zeigt zudem noch die Kosten der Lösung aller Ameisen. Hier erkennt man, dass es in fast jeder Iteration auch Ausreißer nach oben gibt. Dies liegt daran, dass die Ameisen auch weiterhin neue Lösungen erkunden und dabei hin und wieder eine deutlich schlechtere Lösung, als es durchschnittlich der Fall ist, erstellen. Die Graphen der restlichen Parametervariationen sind in Anhang A zu finden.

6.4.4. Vergleich zum greedy Algorithmus

Tabelle 6.6 zeigt die Ergebnisse des greedy Algorithmus im Vergleich zu der optimalen Lösung sowie dem Ameisenalgorithmus. Für das sehr kleine erste Testnetz liefert der greedy Algorithmus ebenfalls das globale Optimum. Doch schon bei dem etwas größerem zweiten Testnetz erreicht der greedy Algorithmus nicht mehr das globale Optimum. Die von ihm konstruierte Lösung liegt mit 22 Kosten bereits 69,2 % über der optimalen Lösung, welche vom Ameisenalgorithmus gefunden wird. Im noch größeren dritten Testnetz erreicht der greedy Algorithmus gerade einmal eine Lösung von 127,9. Das sind 190 % mehr als die optimale Lösung und 113 % mehr als die gefundene Lösung des Ameisenalgorithmus.

Tabelle 6.4.: Ergebnisse der Parameterstudie für das zweite Testnetz

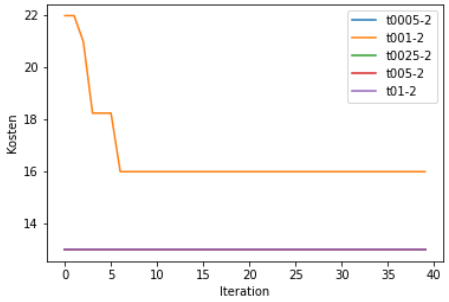
Versuchsname	a1-2	a12-2	a15-2	a2-2	a3-2
globales Optimum gefunden	✓	✓	✓	✓	✓
Versuchsname	b1-2	b12-2	b15-2	b2-2	b3-2
globales Optimum gefunden	✓	✓	✓	✓	✓
Versuchsname	q095-2	q09-2	q085-2	q08-2	q075-2
globales Optimum gefunden	✓	✓	✓	✓	✓
Versuchsname	r01-2	r005-2	r015-2	r02-2	r025-2
globales Optimum gefunden	✓	✓	✓	✓	✓
Versuchsname	x01-2	x005-2	x015-2	x02-2	x025-2
globales Optimum gefunden	✓	✓	✓	✗	✓
Versuchsname	t0005-2	t001-2	t0025-2	t005-2	t01-2
globales Optimum gefunden	✓	✗	✓	✓	✓

Tabelle 6.5.: Ergebnisse der Parameterstudie für das dritte Testnetz

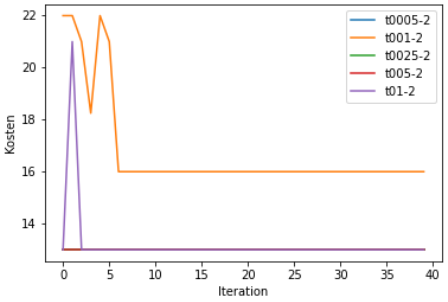
Versuchsname	a1-3	a12-3	a15-3	a2-3	a3-3
günstigste Lösung	69,95	77,95	99,45	73,45	70,95
Versuchsname	b1-3	b12-3	b15-3	b2-3	b3-3
günstigste Lösung	69,95	81,45	69,95	66,00	67,45
Versuchsname	q095-3	q09-3	q085-3	q08-3	q075-3
günstigste Lösung	71,95	69,95	83,45	79,45	65
Versuchsname	r01-3	r005-3	r015-3	r02-3	r025-3
günstigste Lösung	60,00	81,45	61,50	69,95	75,45
Versuchsname	x01-3	x005-3	x015-3	x02-3	x025-3
günstigste Lösung	60,00	82,45	86,90	69,95	68,95
Versuchsname	t0005-3	t001-3	t0025-3	t005-3	t01-3
günstigste Lösung	70,95	69,95	66,45	60,00	80,50

Tabelle 6.6.: Vergleich des greedy Algorithmus mit dem Ameisenalgorithmus

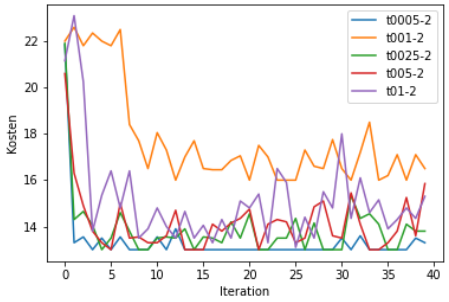
Testnetz	optimale Lösung	Lösung des greedy Algorithmus	Lösung des Ameisenalgorithmus
1	10	10	10
2	13	22	13
3	44	127,9	60



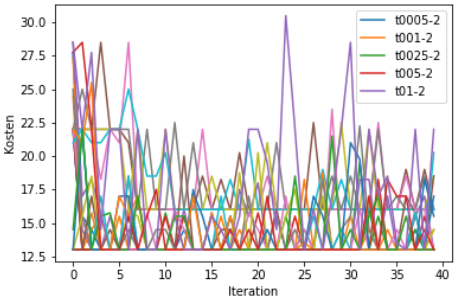
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration



(c) Durchschnittskosten pro Iteration

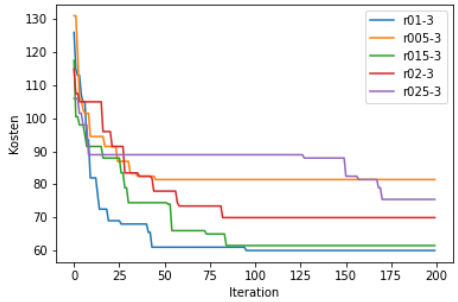


(d) Kosten pro Ameise pro Iteration

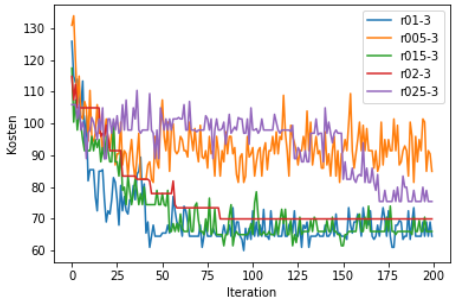
Abbildung 6.6.: Einfluss von τ_0 auf die Kosten von Testnetz 2. Quelle: Eigene Darstellung

6.4.5. Fazit der Ergebnisse

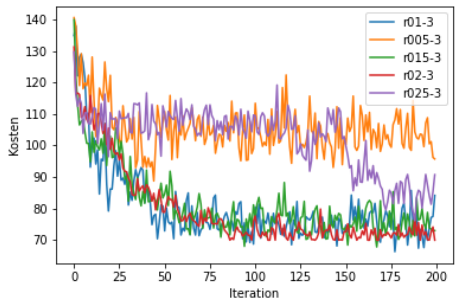
Die Auswertung der Parameterstudie hat gezeigt, dass der Ameisenalgorithmus ein geeignetes Verfahren für das Erstellen von Mittelspannungsnetzen ist. Für kleinere Netze findet er das globale Optimum und bei größeren Testnetzen erreicht er Lösungen, welche in der Nähe des globalen Optimums liegen. Der greedy Algorithmus konnte hingegen nicht überzeugen. Die von ihm erstellten Lösungen lagen bereits bei dem kleineren Netz deutlich über dem globalen Optimum. Bei dem größeren Testnetz lag die von ihm erstellte Lösung sogar fast um den Faktor 3 über der optimalen Lösung.



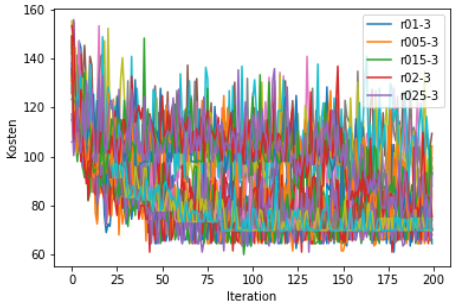
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration



(c) Durchschnittskosten pro Iteration



(d) Kosten pro Ameise pro Iteration

Abbildung 6.7.: Einfluss von ρ auf die Kosten von Testnetz 3. Quelle: Eigene Darstellung

7. Schlussfolgerung und Ausblick

Dieses Kapitel enthält das abschließende Fazit, der gesamten Arbeit. Zudem gibt es einen Ausblick auf mögliche Erweiterungen, sowie einen Implementierungsvorschlag für diese.

7.1. Schlussfolgerung

Die Kombination von Ameisenalgorithmen und einer Triangulierung ist ein geeignetes Verfahren zur Planung von Mittelspannungsnetzen. Durch die Verwendung der Triangulierung konnte der Suchraum verringert werden. Zudem wurde dadurch sichergestellt, dass die Ameisen nur Ringe (bzw. Stränge) erstellen. Die Heuristik der Ameisenalgorithmen stellte sich zudem als geeignet dar. So konnten die Ameisen in zwei Testnetzen das globale Optimum erreichen. In dem anderen Testnetz haben die Ameisen eine gute Lösung erreicht, welche nicht allzu weit vom globalen Optimum entfernt ist. In jedem der drei Fälle haben die Ameisenalgorithmen deutlich besser abgeschnitten als das Vergleichsverfahren. Zudem konnten in einer theoretischen Analyse noch gezeigt werden, dass das Verfahren stets eine Lösung konstruiert, bei der alle Busse angeschlossen werden, sofern die Ringgröße unbeschränkt ist.

7.2. Ausblick

In diesem Kapitel möchte ich einen Ausblick auf mögliche Erweiterungen des Algorithmus geben. Zunächst kann der Algorithmus so erweitert werden, dass er auch die Kapazität berechnet, welche die Leitungen haben müssen. Eine weitere Erweiterung regelt den Umgang mit Leitungen, welche in der Triangulierung enthalten sind, jedoch nicht realisiert werden können. Die letzte hier vorgestellte Erweiterung ist die Berücksichtigung des aktuellen Stromnetzes.

7.2.1. Leitungsausbau

Nach dieser Erweiterung soll der Algorithmus zudem noch ausgeben, welche Leitungen verbaut werden. In der aktuellen Version wird nur ausgegeben wo die Leitungen verlegt werden sollen. Da die Kosten für die Leitungen stark mit den Tiefbaukosten

korrelieren, ist die Auswahl der Leitungen für das Ergebnis nicht wirklich relevant. Es erhöht jedoch die Benutzerfreundlichkeit des Programms enorm, wenn dieser Schritt ebenfalls automatisiert abläuft. Der Grund für die starke Korrelation ist, dass sowohl die Tiefbaukosten als auch die Leitungskosten von der Strecke des Mittelspannungsnetzes abhängen. Die Leitungskosten hängen zudem noch davon ab, wie viel Strom die Leitung maximal transportieren kann. Diese Mehrkosten gegenüber anderen Kabeln sind jedoch im Vergleich zu den Tiefbaukosten und den Basiskosten für Kabel vernachlässigbar. Trotzdem möchte ich im Folgenden noch auf eine Möglichkeit zu der Implementierung eingehen. Sei hierzu ein Ring gegeben, welcher bei einem Transformator T startet und N Busse hat. Der Ring ist hierbei folgendermaßen aufgebaut: $(T, B_1), (B_1, B_2), ..., (B_{N-1}, B_N), (B_N, T)$. Sei V_k die Menge des Stromverbrauchs, welcher an Bus B_k anfällt. Das heißt, der gesamte Stromverbrauch S lässt sich mithilfe der Formel

$$S = \sum_{k=1}^{N} V_k$$

bestimmen. Die Trennstelle, welche im Normalbetrieb geöffnet ist, ist nun idealerweise bei $\frac{S}{2}$. Praktisch startet man am Transformator und geht so lange in eine Richtung (also von B_{a-1} zu B_a), bis $\sum_{k=1}^{a} V_k$ zum ersten mal über $\frac{S}{2}$ ist. An dieser Kante wird die Trennstelle verbaut. Hierbei kann es natürlich zu Extremfällen kommen. Wenn z.B. $V_k = 1000$ ist und alle anderen Busse einen Verbrauch von gerade einmal 1 haben. Dieser Fall tritt in der Praxis jedoch nicht auf, da entweder die anderen Busse einen zu kleinen Verbrauch haben und deshalb an das Niederspannungsnetz angeschlossen werden oder der Bus B_k einen zu großen Verbrauch für das Mittelspannungsnetz hat und deshalb an das Hochspannungsnetz angeschlossen wird. Extremfälle dieser Art müssen also nicht berücksichtigt werden. Wenn man nun die Trennstelle auf diese Art und Weise gefunden hat muss man nun die richtigen Leitungen verwenden. Je nach verwendeter Steuerungsoption (siehe Abschnitt 2.3) unterscheidet sich die maximale Kapazität für die die Leitungen ausgelegt sein müssen. Wenn im Fehlerfall Großunternehmern der Strom abgeschaltet werden darf muss z.B. nicht so viel Puffer freigelassen werden. Nun berechnet man die Summe des Stromverbrauchs S_{Fehler}, welcher im Fehlerfall nicht abgeschaltet werden darf. Falls z.B. die Leitung (T, B_N) ausfällt, wird die Trennstelle geschlossen und der gesamte Strom S_{Fehler} muss durch die Kante (T, B_1). Durch die Kante (B_1, B_2) muss der gesamte Strom S_{Fehler}, abzüglich des Verbrauchs V_1. Für einen Leitung (B_{a-1}, B_a) gilt also, dass sie genügend freie Kapazität haben muss um den Strom $\sum_{k=a}^{N} V_k$ weiterleiten zu können. Natürlich muss auch der symmetrische Fall berücksichtigt werden, in welchem die Kante (T, B_1) ausfällt. In diesem Fall muss die Leitung (B_{a-1}, B_a) genug Puffer haben um den Strom $\sum_{k=1}^{a-1} V_k$ weiterleiten zu können. Die Leitung (B_{a-1}, B_a) muss also für das Maximum dieser beiden Fälle ausgelegt sein. Wenn eine Leitung, außer (T, B_1) oder (T, B_N) kaputt geht benötigen die anderen Leitungen weniger Puffer als in diesen beiden Extremfällen. Deshalb

reicht es diese beiden Fälle zu betrachten. Nun ist klar, wie groß die Leitungskapazität sein muss, damit stets genügend Strom für die Verbraucher weitergeleitet werden kann. Der Fall mit den Einspeisungen lässt sich analog berechnen. Hierbei wird angenommen, dass gerade kein einziger Verbraucher Strom benötigt und die gesamte Energie zum Transformator weitergeleitet werden muss. Auch hierbei erhält man einen Wert für die Leitungsauslastung. Der endgültige Wert für die Kapazität der Leitung ist das Maximum von dem Verbraucherfall und dem Einspeisungsfall.

Um die Leitungsauslastung berechnen zu können muss das übergebene PyPSA-Netz ebenfalls Verbraucher und Erzeuger enthalten. Diese müssen dann von dem Programm in eine passende Form überführt werden. Zudem benötigt das Programm Zugriff auf die möglichen Leitungstypen, welche es verbauen darf.

7.2.2. Nicht realisierbare Leitungen

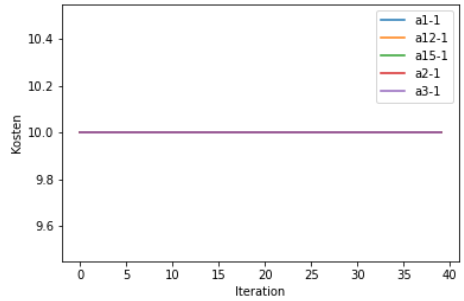
In manchen Fällen ist es nicht möglich zwei Busse miteinander zu verbinden. Einschränkungen dieser Art werden von meinem Algorithmus momentan nicht berücksichtigt. Wenn die Triangulierung eine Kante bauen möchte, welche nicht realisierbar ist, so wird dieser in der aktuellen Version ein 10 % höheres Kantengewicht, als der maximal realisierbaren Kante gegeben. Auf diese Art und Weise versucht der Algorithmus eher dieses Dreieck zu vermeiden. Sollte er das Dreieck doch wählen, so wird er versuchen, das Dreieck so zu erweitern, dass die teure Kante eingespart werden kann. Dieses Verhalten wird durch das verwenden von heuristischen Werten unterstützt, da diese bei größeren Einsparungen sehr hoch sind. In einer zukünftigen Version soll jedoch per Parameter die Option angeboten werden diese Situation anders zu lösen. Man könnte nämlich auch den kürzesten Weg von den beiden Knoten zueinander berechnen, welcher nur realisierbare Leitungen verwendet. Dann wären die Kosten für die nicht realisierbare Leitung, die Kosten für den kürzesten, realisierbaren Weg.

7.2.3. Berücksichtigung des aktuellen Stromnetzes

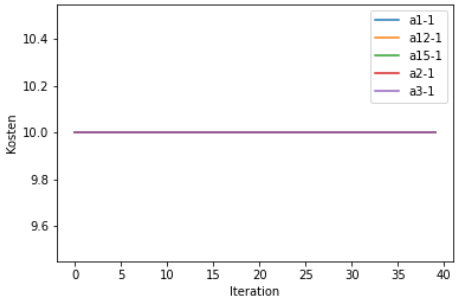
Eine weitere Erweiterung ist die Berücksichtigung des aktuellen Stromnetzes. In diesem erhält der Algorithmus als Eingabe zusätzlich alle existierenden Leitungen. Nun werden die Kosten für die Entfernung aller Leitungen berechnet. Die Gesamtkosten sind dann die Kosten für das neue Stromnetz und den Rückbau der nicht mehr benötigten Leitungen. Wenn die Ameise in der Erstellung der Lösung eine Leitung bauen möchte, welche bereits existiert, so fließen diese Kosten negativ in die Gesamtkosten ein, da dann der Rückbau für diese Leitung eingespart wird.

A. Vollständige Ergebnisse der Parameteranalyse

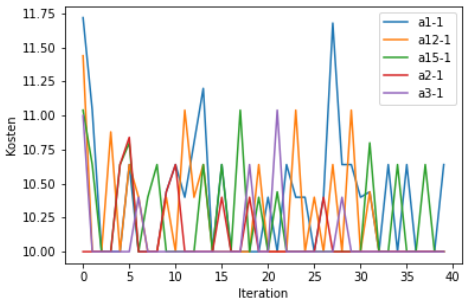
Auf den folgenden Seiten sind die ausführlichen Ergebnisse der Parameterstudie zu finden. Eine Diskussion dieser Ergebnisse ist in Kapitel 6 zu finden.



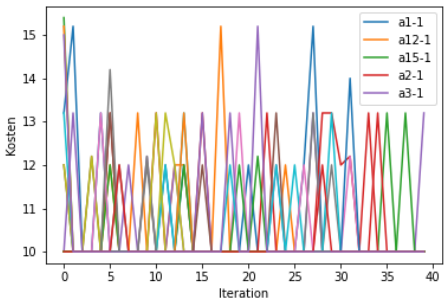
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration

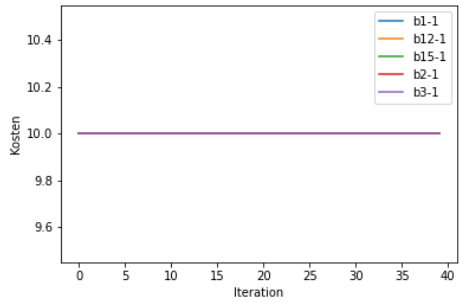


(c) Durchschnittskosten pro Iteration

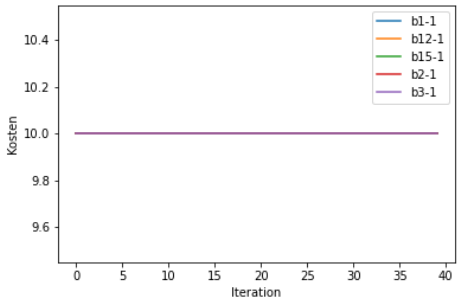


(d) Kosten pro Ameise pro Iteration

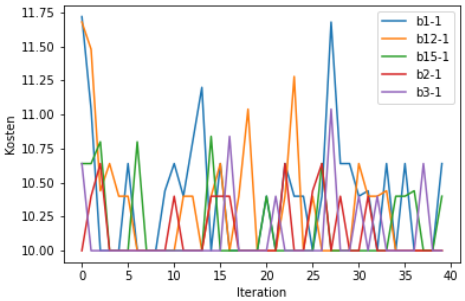
Abbildung A.1.: Einfluss von α auf die Kosten von Testnetz 1. Quelle: Eigene Darstellung



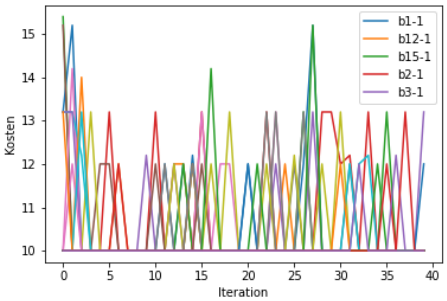
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration

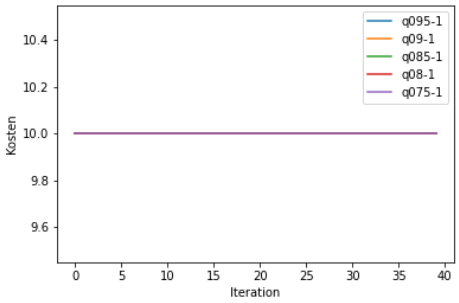


(c) Durchschnittskosten pro Iteration

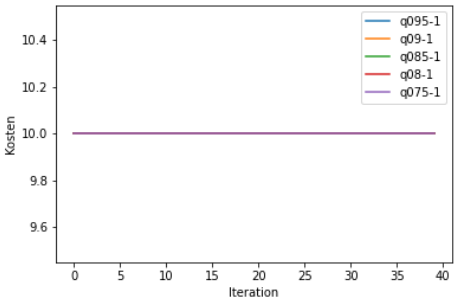


(d) Kosten pro Ameise pro Iteration

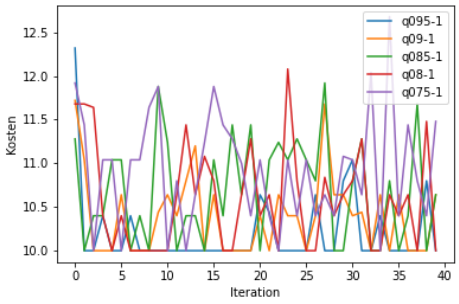
Abbildung A.2.: Einfluss von β auf die Kosten von Testnetz 1. Quelle: Eigene Darstellung



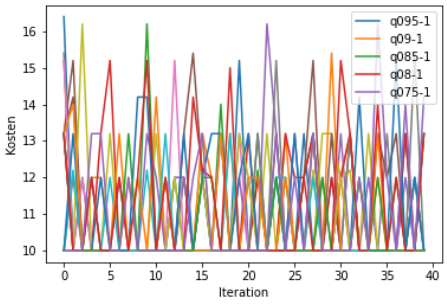
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration

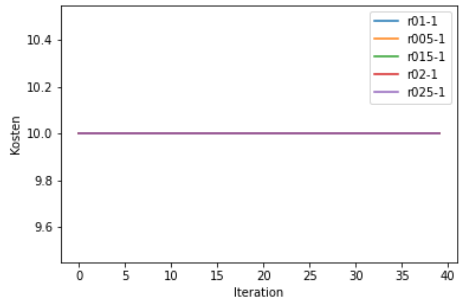


(c) Durchschnittskosten pro Iteration

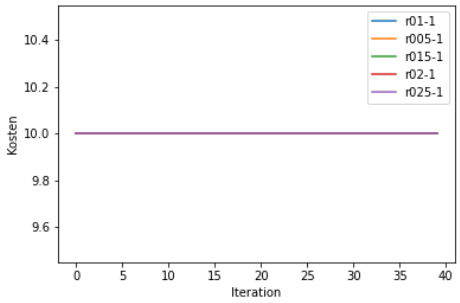


(d) Kosten pro Ameise pro Iteration

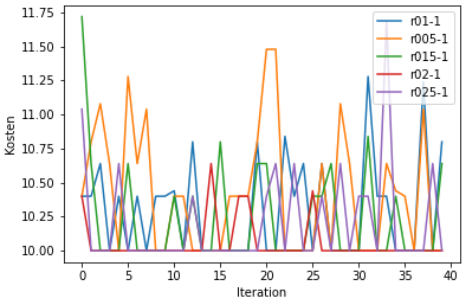
Abbildung A.3.: Einfluss von q_0 auf die Kosten von Testnetz 1. Quelle: Eigene Darstellung



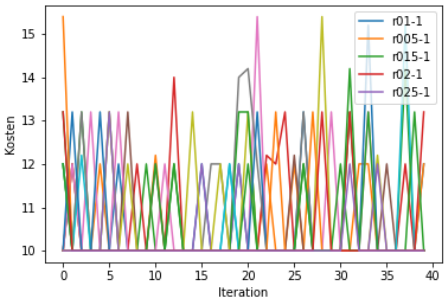
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration

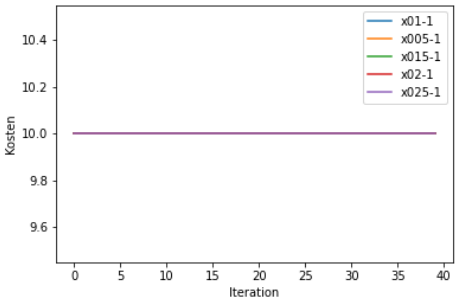


(c) Durchschnittskosten pro Iteration

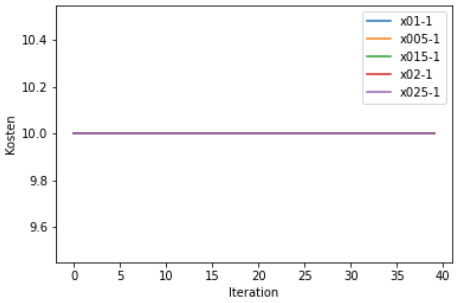


(d) Kosten pro Ameise pro Iteration

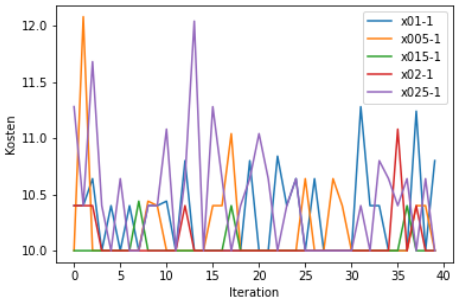
Abbildung A.4.: Einfluss von ρ auf die Kosten von Testnetz 1. Quelle: Eigene Darstellung



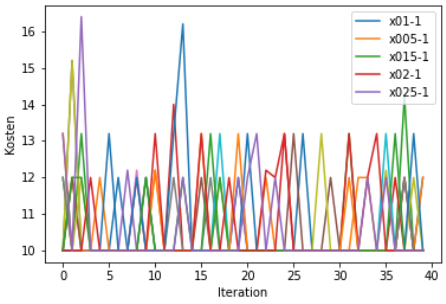
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration

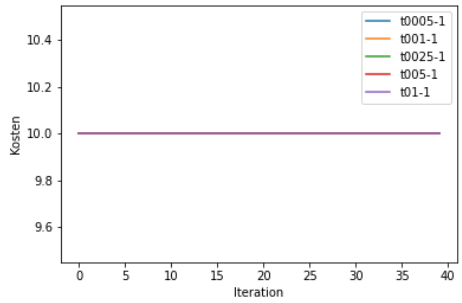


(c) Durchschnittskosten pro Iteration

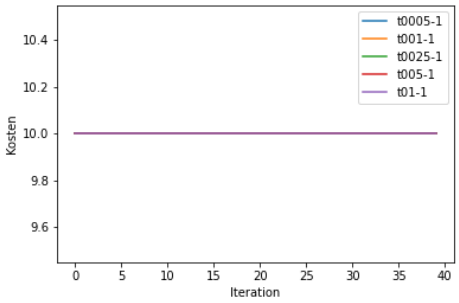


(d) Kosten pro Ameise pro Iteration

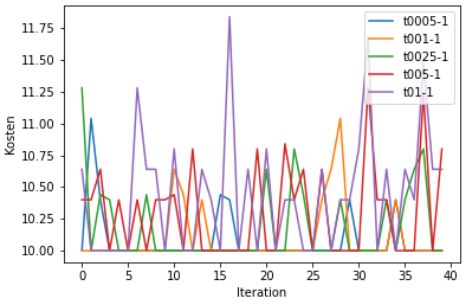
Abbildung A.5.: Einfluss von ξ auf die Kosten von Testnetz 1. Quelle: Eigene Darstellung



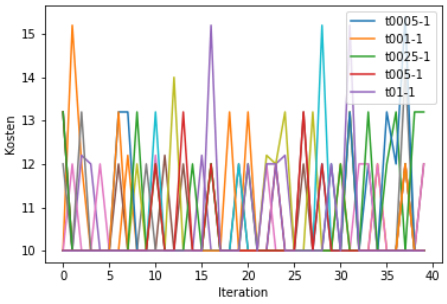
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration

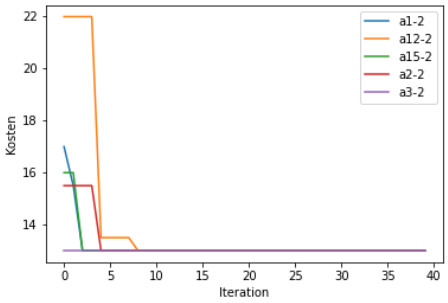


(c) Durchschnittskosten pro Iteration

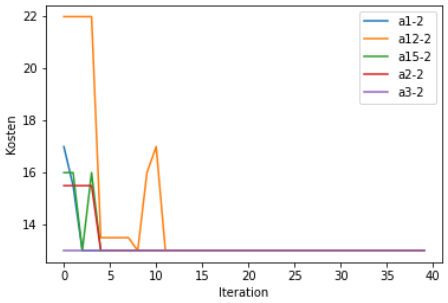


(d) Kosten pro Ameise pro Iteration

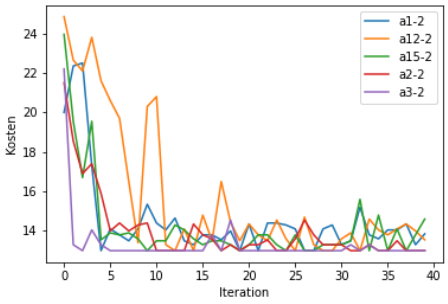
Abbildung A.6.: Einfluss von τ_0 auf die Kosten von Testnetz 1. Quelle: Eigene Darstellung



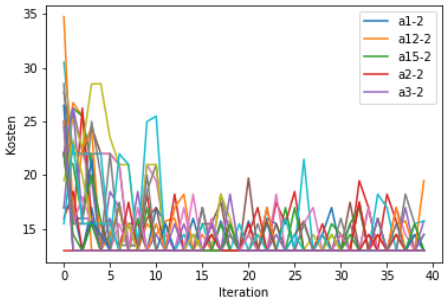
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration

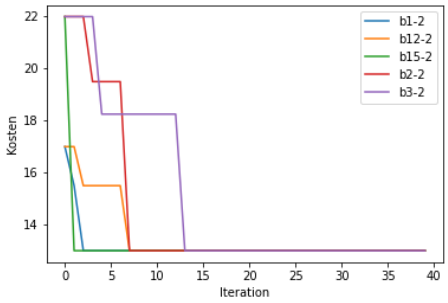


(c) Durchschnittskosten pro Iteration

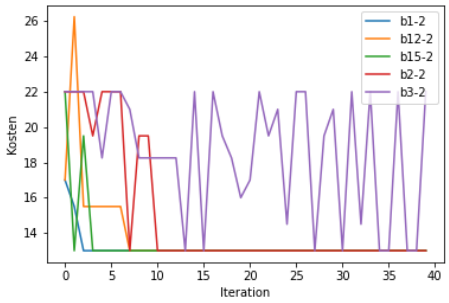


(d) Kosten pro Ameise pro Iteration

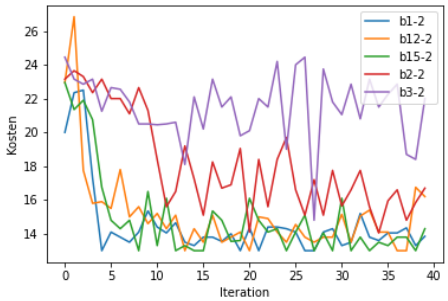
Abbildung A.7.: Einfluss von α auf die Kosten von Testnetz 2. Quelle: Eigene Darstellung



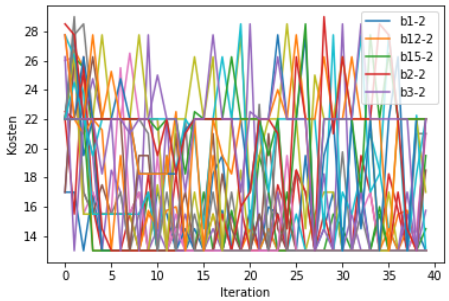
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration

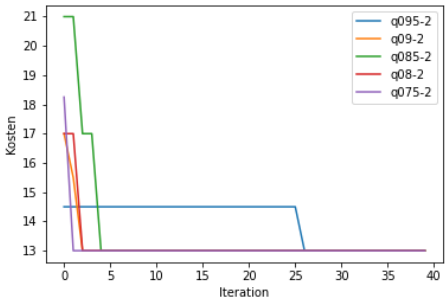


(c) Durchschnittskosten pro Iteration

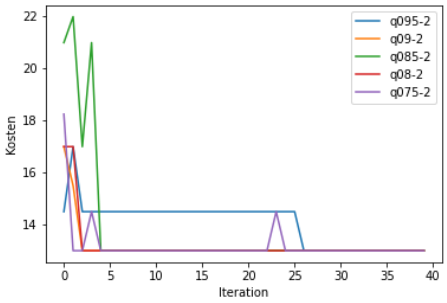


(d) Kosten pro Ameise pro Iteration

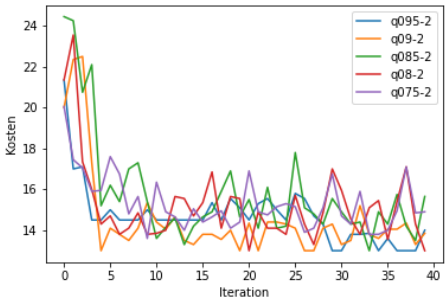
Abbildung A.8.: Einfluss von β auf die Kosten von Testnetz 2. Quelle: Eigene Darstellung



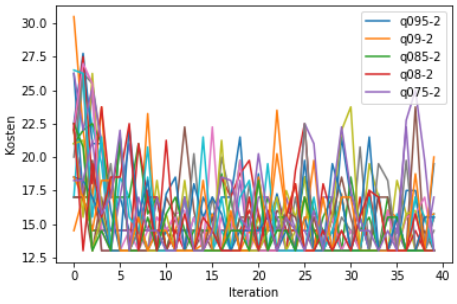
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration

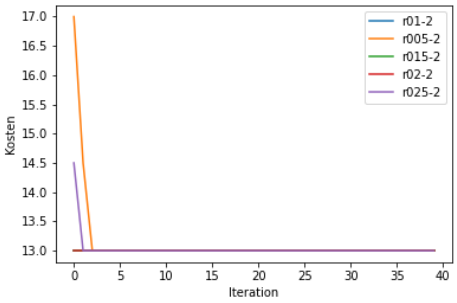


(c) Durchschnittskosten pro Iteration

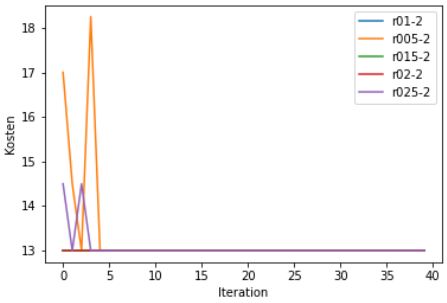


(d) Kosten pro Ameise pro Iteration

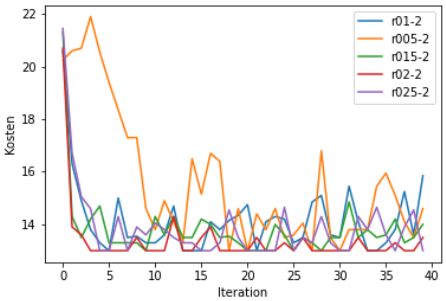
Abbildung A.9.: Einfluss von q_0 auf die Kosten von Testnetz 2. Quelle: Eigene Darstellung



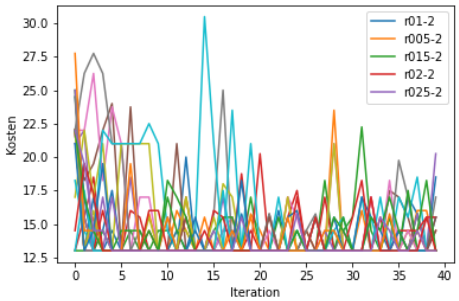
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration

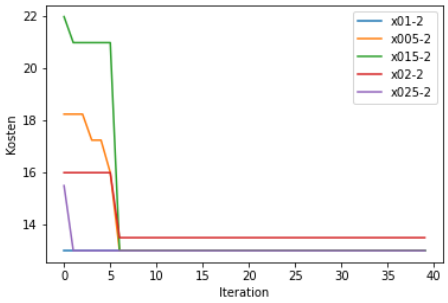


(c) Durchschnittskosten pro Iteration

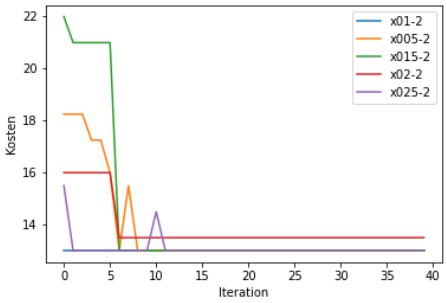


(d) Kosten pro Ameise pro Iteration

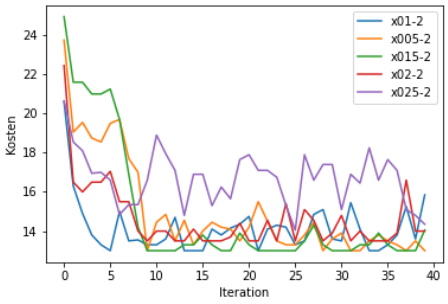
Abbildung A.10.: Einfluss von ρ auf die Kosten von Testnetz 2. Quelle: Eigene Darstellung



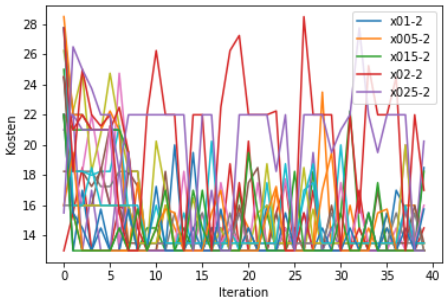
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration

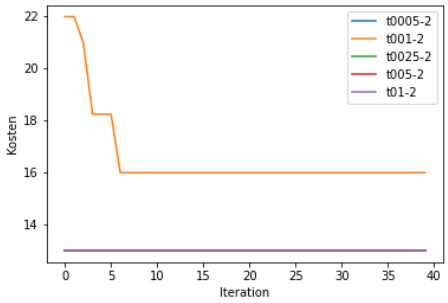


(c) Durchschnittskosten pro Iteration

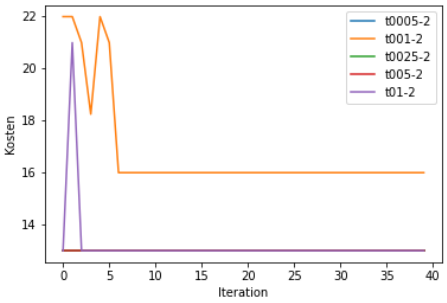


(d) Kosten pro Ameise pro Iteration

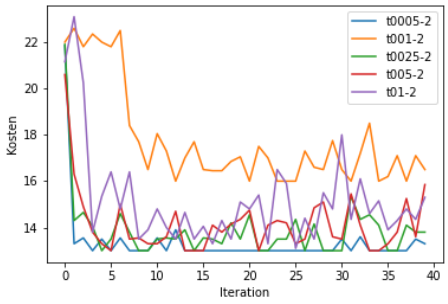
Abbildung A.11.: Einfluss von ξ auf die Kosten von Testnetz 2. Quelle: Eigene Darstellung



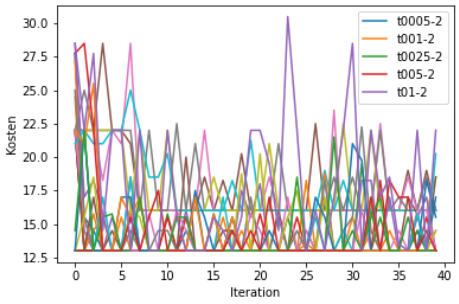
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration

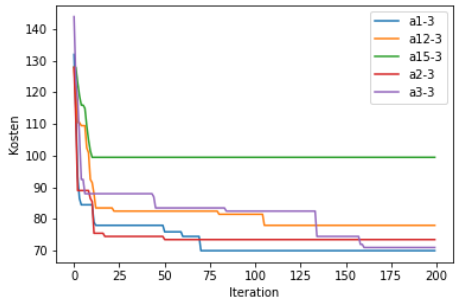


(c) Durchschnittskosten pro Iteration

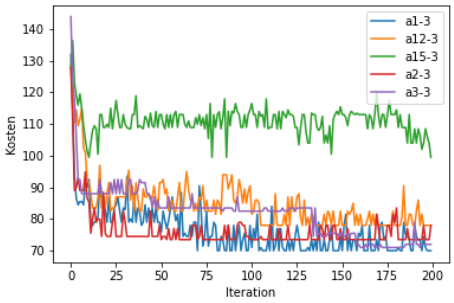


(d) Kosten pro Ameise pro Iteration

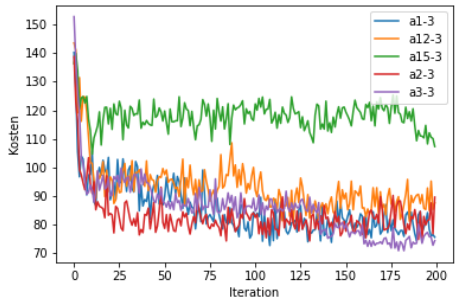
Abbildung A.12.: Einfluss von τ_0 auf die Kosten von Testnetz 2. Quelle: Eigene Darstellung



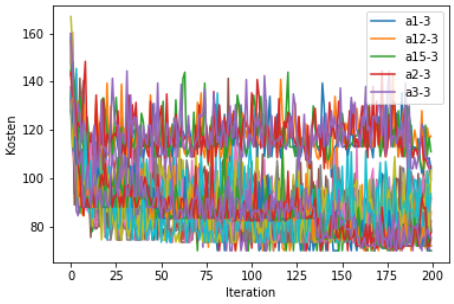
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration

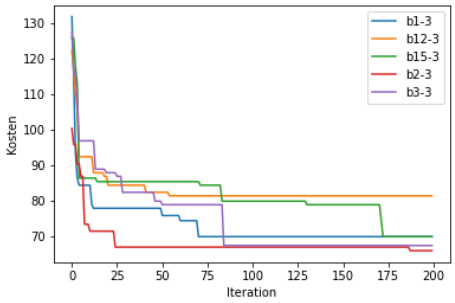


(c) Durchschnittskosten pro Iteration

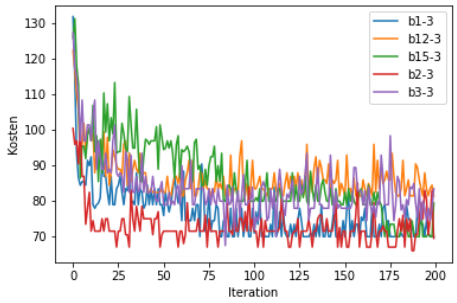


(d) Kosten pro Ameise pro Iteration

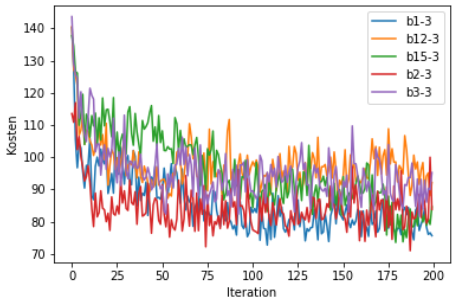
Abbildung A.13.: Einfluss von α auf die Kosten von Testnetz 3. Quelle: Eigene Darstellung



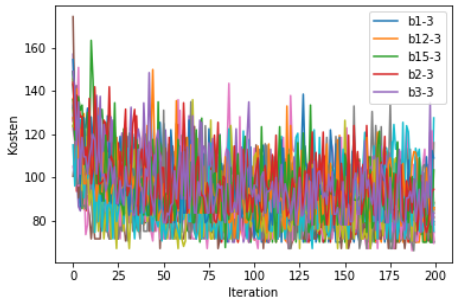
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration

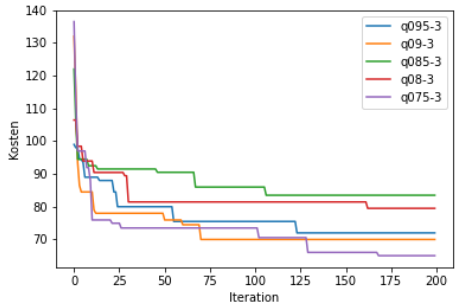


(c) Durchschnittskosten pro Iteration

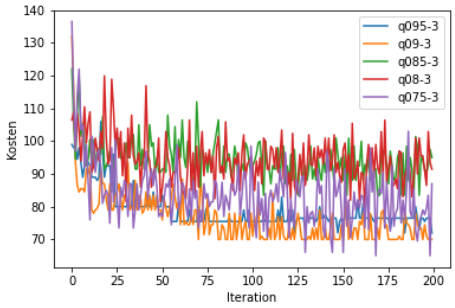


(d) Kosten pro Ameise pro Iteration

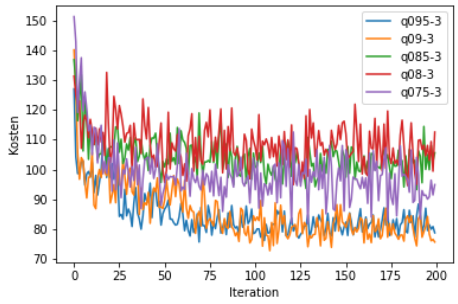
Abbildung A.14.: Einfluss von β auf die Kosten von Testnetz 3. Quelle: Eigene Darstellung



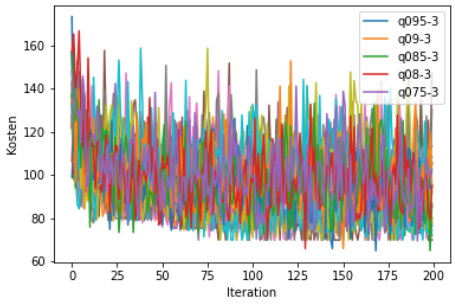
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration

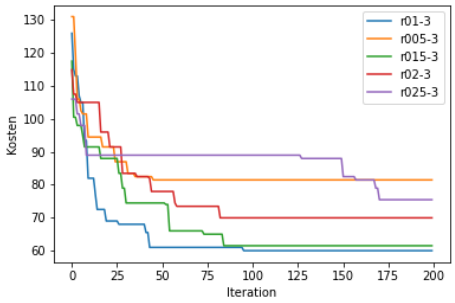


(c) Durchschnittskosten pro Iteration

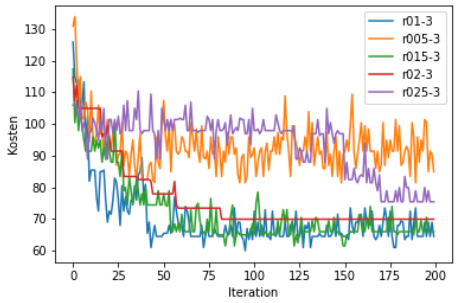


(d) Kosten pro Ameise pro Iteration

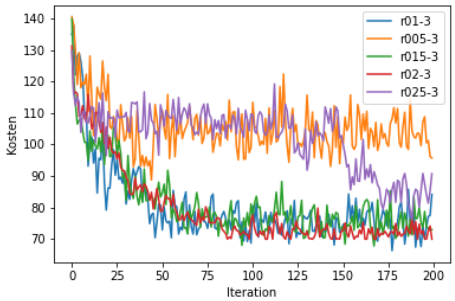
Abbildung A.15.: Einfluss von q_0 auf die Kosten von Testnetz 3. Quelle: Eigene Darstellung



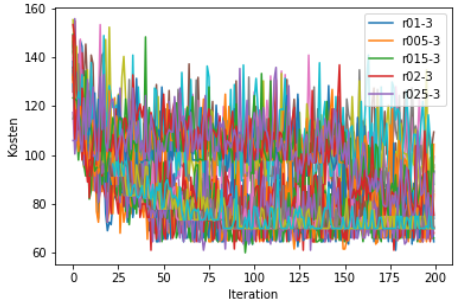
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration

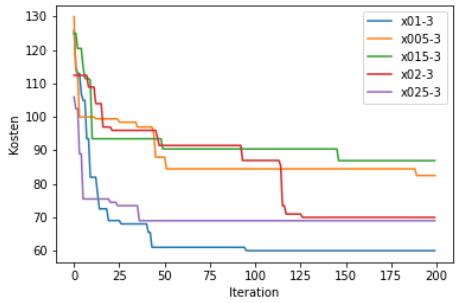


(c) Durchschnittskosten pro Iteration

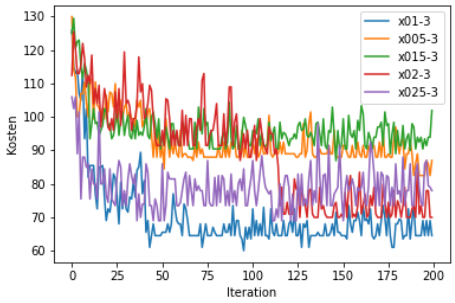


(d) Kosten pro Ameise pro Iteration

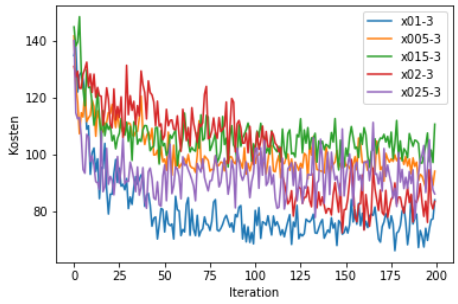
Abbildung A.16.: Einfluss von ρ auf die Kosten von Testnetz 3. Quelle: Eigene Darstellung



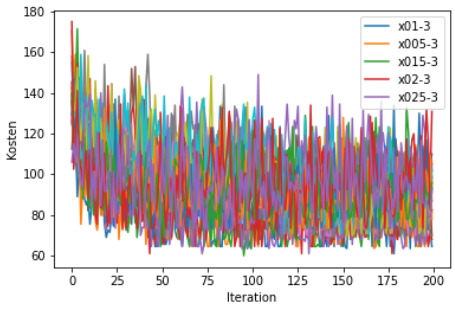
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration

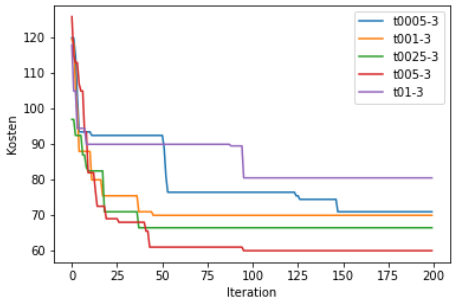


(c) Durchschnittskosten pro Iteration

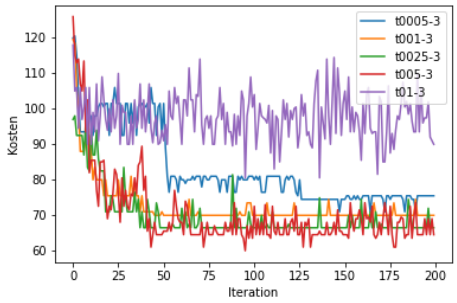


(d) Kosten pro Ameise pro Iteration

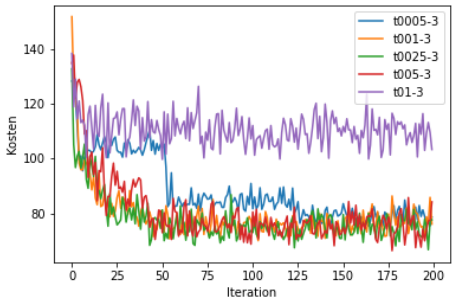
Abbildung A.17.: Einfluss von ξ auf die Kosten von Testnetz 3. Quelle: Eigene Darstellung



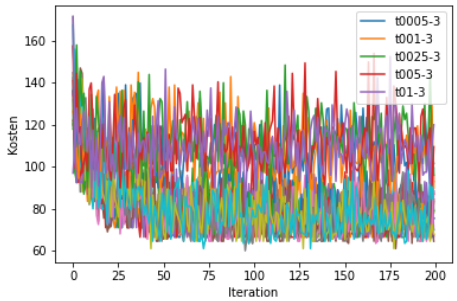
(a) Minimale Kosten seit Beginn



(b) Minimale Kosten pro Iteration



(c) Durchschnittskosten pro Iteration



(d) Kosten pro Ameise pro Iteration

Abbildung A.18.: Einfluss von τ_0 auf die Kosten von Testnetz 3. Quelle: Eigene Darstellung

Literaturverzeichnis

[1] Bundesregierung, “Was tut die Bundesregierung für den Klimaschutz?” [Online]. Available: https://www.bundesregierung.de/breg-de/themen/klimaschutz/bundesregierung-klimapolitik-1637146

[2] Bundesministerium für Wirtschaft und Energie, “Unsere Energiewende: sicher, sauber, bezahlbar.” [Online]. Available: https://www.bmwi.de/Redaktion/DE/Dossier/energiewende.html

[3] Bundesministerium für Wirtschaft und Energie, “Erneuerbare Energien.” [Online]. Available: https://www.bmwi.de/Redaktion/DE/Dossier/erneuerbare-energien.html

[4] Bundesministerium für Wirtschaft und Energie, “Energiewende-Plattform Energienetze.” [Online]. Available: https://www.bmwi.de/Redaktion/DE/Artikel/Energie/energiewende-plattform-energienetze.html

[5] Bundesministerium für Wirtschaft und Energie, “Ein Stromnetz für die Energiewende.” [Online]. Available: https://www.bmwi.de/Redaktion/DE/Dossier/netze-und-netzausbau.html

[6] Niklas Rotering, “Zielnetzplanung von Mittelspannungsnetzen unter Berücksichtigung von dezentralen Einspeisungen und steuerbaren Lasten,” Ph.D. dissertation, rheinisch-westfälische Technische Hochschule Aachen, 2012.

[7] Viktoria Neimane, “On development planning of electricity distribution networks,” Ph.D. dissertation, KTH Royal Institute of Technology, 2001.

[8] Marco Dorigo und Thomas Stützle, *Ant Colony Optimization*. Massachusetts Institute of Technology, 2004.

[9] Stefan Riepl, “Stromversorgung.svg.” [Online]. Available: https://de.wikipedia.org/wiki/Datei:Stromversorgung.svg

[10] “Self-organized Shourtcuts in the Argentine Ant,” *Naturwissenschaften*, no. 76, pp. 579–581, 1989.

[11] Lukas Gebhard, “Expansion Planning of Low-Voltage Grids Using Ant Colony Optimization,” Master’s thesis, Albert Ludwigs Universität Freiburg, 2021.

[12] Thomas Brown, Jonas Hörsch und David Schlachtberger, “PyPSA: Python for Power System Analysis,” *Journal of Open Research Software*, vol. 6, no. 4, 2018. [Online]. Available: https://doi.org/10.5334/jors.188

[13] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[14] Aric A. Hagberg and Daniel A. Schult and Pieter J. Swart, "Exploring Network Structure, Dynamics, and Function using NetworkX," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.

[15] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[16] Kirill Simonov, "PyYAML." [Online]. Available: https://pyyaml.org/