Master's Thesis

Ant Colony Optimization for Medium Voltage Grids

Paul Willi

Examiner: Prof. Dr. Christian Schindelhauer
Advisers: Wolfgang Biener, Janis Kähler

University of Freiburg
Technical Faculty
Department of Computer Science

September 12th, 2022

Writing Period

26. 04. 2022 – 12. 09. 2022

Examiner

Prof. Dr. Christian Schindelhauer

Chair of Computer Networks and Telematics

Second Examiner

Prof. Dr. Christof Wittwer

Director of Division Power Electronics, Grids and Smart Systems at Fraunhofer ISE

Advisers

Wolfgang Biener, Janis Kähler

Declaration

I hereby declare that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Place, Date	Signature

Abstract

The transition to a carbon neutral energy supply is one of the greatest challenges of our time. To enable the use of renewable energy sources, a well-developed energy transmission system plays an important role. However, the planning of grid expansion gets more difficult with increasingly decentralized and volatile electricity generation. Therefore, scientists developed automated planning tools to assist grid operators in planning grid expansion. To complement previous research, this work presents *Ant-Power-Medium-Voltage*, an algorithm for target network planning of medium voltage grids via Ant Colony Optimization, a heuristic optimization method. Furthermore, the presented algorithm is tested and evaluated on a real world example and manages to find a solution which reduces the cost by 11%. A parameter study is conducted to show the influence of the algorithms input parameters. Although limitations of the presented algorithm are elaborated, this work shows that Ant Colony Optimization is a promising tool to improve automated planning of electric grids.

Contents

List of Figures

List of Algorithms

1. Introduction

The Climate crisis requires a rapid transition from fossil to renewable energy sources to reduce carbon emissions. In comparison to conventional power plants, renewable energy resources like wind and solar are placed more decentralized across the electric grid and produce energy in less predictable patterns. Additionally, the demand for electricity will increase drastically in the upcoming years notably due to the widespread usage of heat pumps and electric vehicles [1]. This poses new challenges on the grid to stably deliver electricity from the producer to the consumer. To cope with the additional requirements, the power grid must be expanded in many places. According to a report filed by the federal network agency of Germany in 2021, the cost of the network expansion for the coming ten years amounts to 15.84 billion Euros, of which 7.86 billion Euros are estimated for middle voltage (MV) grid expansion [2]. The development of tools for smart MV grid expansion is therefore not only relevant for the future supply of electricity but also for public and private funding.

However, the planning of distribution grid expansion gets increasingly complex, due to the new requirements mentioned above. This makes is it more difficult for experts to rely on manual planning only. Instead, distribution system operators increasingly use automated planning methods for assistance. [3] categorizes network planning into *target network planning* and *expansion planning*, depending on the applied time horizon. *Target network planning* has a scope of a longer time horizon (more than ten years) and does not take existing grid infrastructure into account. It assumes that a new grid can be built from scratch and tries to find an optimal solution which

fulfills all given requirements. This is typically used for the planning of developing areas and long-term grid transformations. *Network expansion planning* on the other hand takes the existing grid infrastructure into account and tries to modify it in a way that it can cope with the short- and mid-term demands (time horizon five to ten years). Generally, *target planning* is done in a first step and afterwards the existing network is modified using *expansion planning* in the direction of the target network solution. In this work network planning generally refers to *target network planning*. Fo transmission efficiency reasons the grid is separated into a high-, medium- and low-voltage level. Each of them with different specific requirements which an automated planning tool must consider. A special characteristic of the medium-voltage level is, for example, the *n-1 criterion*, according to which the supply to each station must be guaranteed despite the failure of one element. It therefore stands to reason that an automated planning tool would be developed for a specific voltage level. The presented work addresses the problem of medium voltage grid planning via Ant Colony Optimization (ACO). ACO is a heuristic optimization method which performed well on grid planning problems in the past [3].

This work is structured in the following way. Chapter 2 gives an overview of the research in the field. Chapter 3 provides background information on electric grids, an introduction to ant colony optimization and how it can be used in the context of grid planning. Chapter 4 formulates the optimization problem and follows with the presentation of *Ant-Power-Medium-Voltage (APMV)*, an algorithm for the planning of MV grids. Subsequently, the algorithm is tested and evaluated on a real-world example grid in Chapter 5. Chapter 6 concludes this work with an outlook on future research.

2. Related Work

From the introduction in the previous chapter it is clear that the expansion of the electric grid is often necessary to cope with the growing demand for electricity in context of the energy transition. In the literature exist different approaches for power distribution planning [4]. They vary strongly in the methods they use, from exact numeric methods like non-linear- or dynamic-programming [5, 6] to heuristic methods like genetic algorithms [7], tabu search [8] and ant-colony-optimization. In addition to the different methods, they also differ in terms of side constraints which have to be fulfilled, as well as assumptions about the nature of the grid and inputs which need to be provided. Numeric methods are attractive because they produce exact and optimal solutions. Unfortunately, their performance is often times not good enough for most practical applications. Many challenges of grid planning involve NP-complete problems (e.g. related to the Traveling Salesman Problem) which leads to very long computational time of numeric methods even for strong machines. To reduce the run-time they need to simplify the problem to a great extend such that the results can not easily be translated to real world applications anymore. Due to this reason, heuristic methods seem to be more promising since they can find acceptable solutions even for NP-complete problems in polynomial time. This speed advantage makes them also applicable to more realistic scenarios without oversimplification. Unfortunately, heuristics can also introduce inaccuracies, which could lead to overconfidence in the produced solution. A thorough testing of the algorithm and an evaluation of the results is therefore necessary. This work uses a heuristic Ant Colony Optimization algorithm for network planning since they

showed promising results in the past [9][10][11].
Four sources were especially influential for the creation of this work and are presented in more detail in the upcoming sections. Afterwards, a brief overview of the new contributions to grid planning by this work is given.

2.1. Dorigo et. al.

Ant colony optimization methods are known to perform well on many grid planning problems. They were first introduced by Dorigo et. al. in 1996 in an algorithm called *Ant System* [9]. *Ant System* was tested and evaluated on the famous NP-complete Traveling Salesman Problem (TSP). Inspired by real ants astonishing ability of swarm intelligence to find short routes to sources of food Dorigo et. al. used artificial ants which can communicate and learn through pheromone trails left behind by ants of previous iterations. The basic idea is that shorter paths to the food source will accumulate a higher density of pheromones, which leads to a convergence of most ants towards the shorter path. The artificial ants construct a technically valid solution and in a global pheromone update step the best solution is rewarded by an increase in pheromones whereas the other solutions are punished by a loss of pheromones.

One year later in 1997 Dorigo et. al. published an improved version called *ant colony system* [11] which uses a local pheromone update in addition to the global pheromone update already used in the original algorithm. This local update removes pheromone on the paths already used by the ant, such that the next ant is more likely to chose a different path. This leads to a larger part of the search space being explored and reduces that chance of convergence towards a local optimum. The same procedure is also used in the algorithm of this work. A more detailed introduction to ant colony optimization algorithms with examples can be found in Chapter 3.2.

2.2. Gebhard

In 2021 Gebhard developed an algorithm in his thesis that uses ant colony optimization for expansion planning of low voltage grids called *AntPower* [12]. *AntPower* combines conventional power line expansion with a reconfiguration of the power switches. Gebhard provides a great introduction into the planning of grids and the technical foundations for scholars outside electrical engineering like mathematics or computer science. Additionally, he gives a detailed description into how ACO is implemented in the context of grid planning. To evaluate the algorithm, real data of a village in rural Germany is used. His results show that the expansion plan generated by *AntPower* is 60% cheaper than an expansion plan obtained through conventional, manual planning based on expert knowledge and 64% cheaper than the expansion plan generated using a local search algorithm.

2.3. Rotering

For the optimization of medium voltage grids via ACO, the dissertation of Rotering [3] gives a great overview. Rotering developed a procedure for medium voltage target planning using ant colony optimization which is also able to consider controllable loads and generators. In addition to network costs he also conducted an economic evaluation, which is part of the network design, but methodologically separate from the energy supply task of the network.

Rotering translates the electric grid into a graph and points out similarities between the medium voltage network problem and other optimization problems like the TSP. Since Ant Colony algorithms were specifically developed for the TSP Rotering opted to use them as an optimization tool. A crucial step in his procedure is the reduction of the potential connections between local network stations. He only considers edges of triangles created by the Delaunay triangulation as potential lines between networks stations. This dramatically reduces the search space and therefore reduces

the runtime of the algorithm (further details in 3.3.1). In addition, a triangle in itself already constitutes a ring structure which is important for the topological constraints of a MV grid (*n-1 criterion*). Like Dorigo et. al. presented in [11], Rotering uses a local and a global pheromone update for his algorithm. After the construction of a legal network design Rotering uses additional optimization methods to further improve the solution. He evaluates the his procedure on two MV grids and receives promising results.

2.4. Zeller

Zellers thesis about the planning of medium voltage grids via ACO serves as the basis for this work. The presented ACO algorithm *APMV* is s a further development of the algorithm Zeller presents in [13]. Zeller already uses some concepts like the Delaunay triangulation, a local and global pheromone update and pheromone evaporation in his algorithm. Zeller also considered special cases originating from the triangulation which could lead to faulty networks (called *back-and-forth* cases). They are explained in further detail in 4.2.6. In these cases Zeller extended and optimized the procedure to build valid solutions. Besides a practical analysis and a parameter study he conducted a theoretical analysis and provided several proofs to further legitimize the procedure. He showed, that given enough runtime the algorithm would always find the global optimum and that all buses would be connected. Additionally Zeller showed that the topological constraints would always be fulfilled. For evaluation of the developed algorithm Zeller used manually built synthetic grids with sizes of 5 to 30 nodes.

2.5. New contributions

This work contributes to grid planning via ACO in the following ways:

1. An improved algorithm for optimizing medium voltage grids called *Ant-Power-Medium-Voltage* is presented.
2. The algorithm is tested and evaluated on a real world cross-voltage grid.
3. Cases, which previously lead to faulty network structures are intercepted.
4. A parameter study to find the algorithms best performance is conducted.

Regarding grid planning, so far in the literature many methods for grid planning are provided without testing and evaluating them on real world examples. Scientific research however, is only useful if it can be converted into an improvement of the practical application. Therefore, the aim of this work is to use the algorithm to optimize a real world cross-voltage grid and evaluate its performance. This grid is taken from a village which is located in rural Germany and and resembles a typical application example.
Besides applying existing research on a real world example, this work also presents an ACO algorithm with several improvements. Firstly, the accuracy of the cost function is increased by several measures. Secondly, the laying of new lines is done alongside streets to make more realistic estimations. Thirdly, overlapping rings in the network are handled in a sensible way. And lastly, cases which previously lead to faulty network structures are intercepted.

3. Background

The upcoming chapter gives a basic description of electric grids and their structure. Afterwards, the functionality of ACO and its application to grid planning is presented. The goal is to provide sufficient background information for the understanding of the presented algorithm and its evaluation (i.e. for scholars of computer science or mathematics without previous knowledge about electric grids).

3.1. Electric Grids

An electric grid is the infrastructure, which enables the distribution of electric energy. Important components of a grid are the following.

3.1.1. Components

Sources: At sources electricity is fed into the electric grid (power plants, wind turbines, solar-panels etc.).

Loads: At loads electricity is subtracted from the grid (households, factories etc.).

Lines: Transmission lines connect sources and loads and transport the electricity.

Buses: A bus is a connection point between two lines or more. Loads and sources can be coupled to a bus but it can also be a mere connection point.

Transformers: Transform the voltage from one voltage level to another. A transformer can therefore also be seen as a generator or source within a certain voltage level. It functions as a connection between the different voltage levels of the grid.

Switches: Enable the opening and closing of connections between different branches of the grid. They are used cut off electricity from a branch in case of an emergency or to supply electricity to the consumer via an alternative route (n-1 criterion).

Storage Units: Enable the storage of energy (via pumped storage hydroelectricity, batteries, hydrogen etc.).

Transformers: Transform the voltage from one voltage level to another one. A transformer therefore can also be seen as a generator or source from within a certain voltage level.

(**Slack bus**: In power flow studies, the slack bus is used to provide for system losses by emitting or absorbing active and reactive power to and from the system. It is usually located at the transformer connecting to the higher voltage level.)

3.1.2. Structure

To increase efficiency, the grid is usually separated into three different voltage levels: high-(HV), medium-(MV) and low-voltage (LV).

HV grids connect big power plants and transformer stations of underlying medium-voltage (MV) grids. They usually operate at voltages reaching from 110 to 380 kV. Their structure is more meshed to build up redundancy for safety.

MV grids connect smaller power plants, factories and transformer stations to the low-voltage (LV) level. They operate at around 10 to 20 kV and are required to fulfill the *n-1 criterion*. So if one component of the MV grid fails, there is at least one built in redundancy to keep the grid functional.

LV grids mostly transfer electricity from the transformer stations to the households. Due to the increasing decentralized feedin of electricity the LV grid also needs to be capable of transferring energy between households. They usually operate at voltages of 0.4 kV.

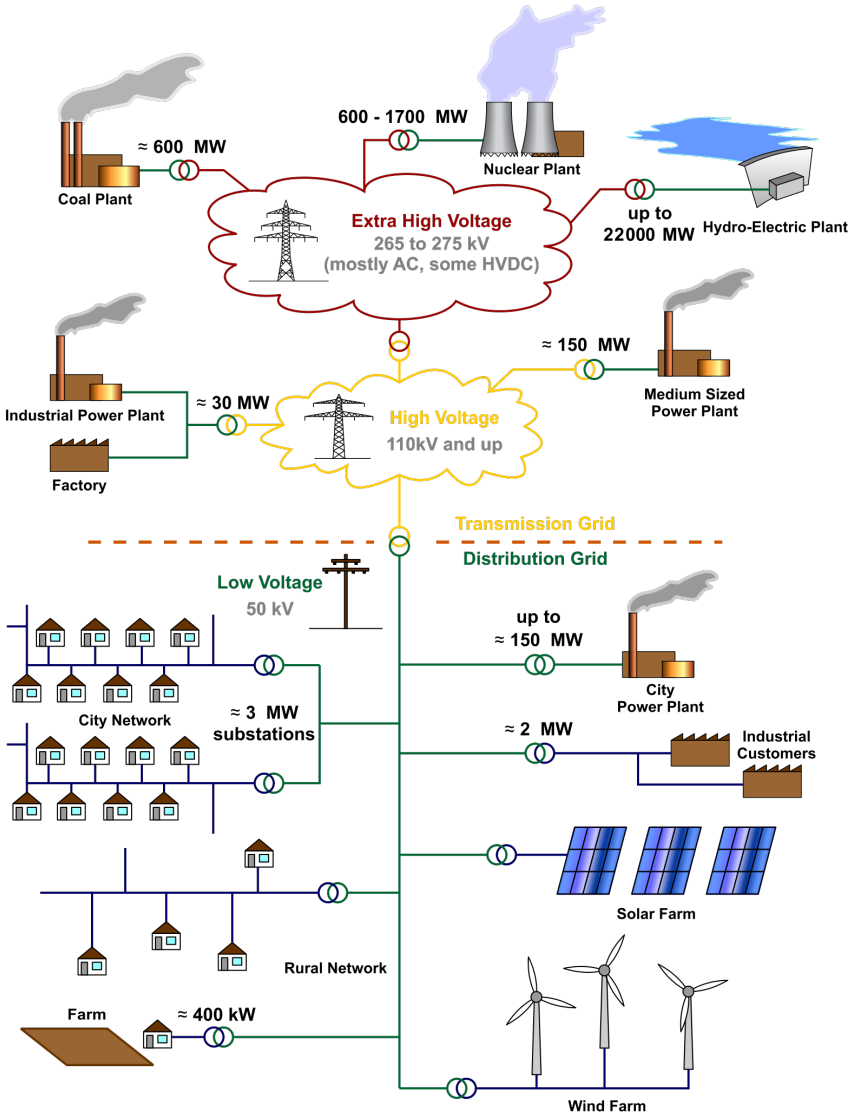

Figure 1.: Schematic depiction of the electricity grid common in Germany. Source: https://commons.wikimedia.org/wiki/File:Electricity_Grid_Schematic_English.svg

3.2. Ant Colony Optimization

ACO is inspired by the astonishing ability of real ants to find short paths between their nest to sources of food. Ants are mostly blind and coordinate their movements using pheromones, which they leave behind on the trails they use. Ants are able to sense the pheromones and tend to follow paths with a higher concentration of the messenger substance. Ants, which probabilistically follow the paths with higher pheromone concentration, again leave behind pheromones, which increase the chance of other ants to also follow this trail. Over time, this leads to a convergence of ants to follow the path with the highest pheromone concentration.

3.2.1. The Double Bridge Experiment

The following example is based on experiments Deneubourg et al. conducted with Argentine ants in 1990 [14]. The ants are placed at the starting location and two paths of different length connect them to a source of food. As shown in 2, at the beginning of the experiment the ants decide randomly which path they choose, since no pheromone has been distributed yet (a). Because one path is shorter, the ants who chose this path arrive at the food source earlier (b). On the way back, there only exists pheromone on the shorter path, which leads to a higher chance of ants choosing it (c). Over time, this leads to the convergence of most ants following the short path (d).

In general, ants do not deterministically follow the path with the highest pheromone concentration, but just with a higher chance. This means, that there is still the possibility of an ant to chose a different path and explore new routes. Thereby, new sources of food or potentially even shorter routes can be discovered. However, the discovery of new short paths alone is not enough to get the majority of ants to switch to them. In the real world, but also for ACO the pheromones are evaporating over time, which leads to the "forgetting" of old established routes. On shorter paths, the

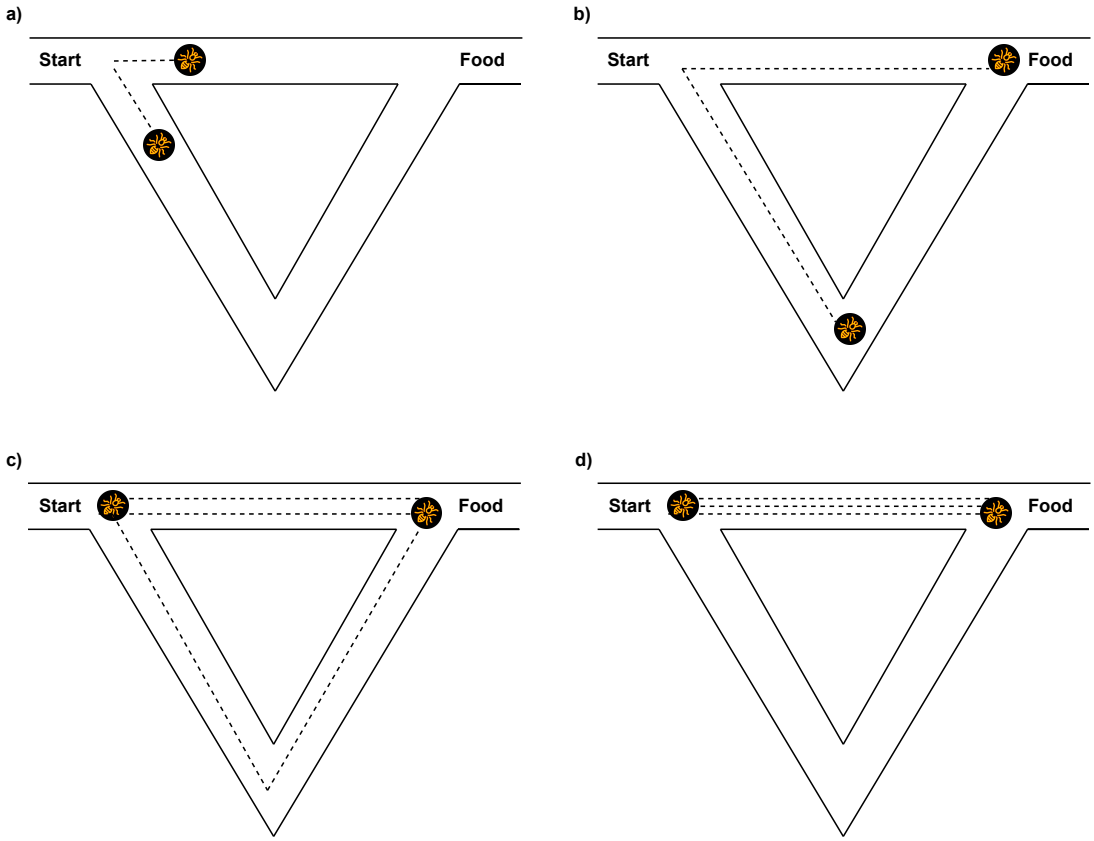

Figure 2.: Double Bridge Experiment. Two paths of different length lead from the start to the food source.

ants refresh the pheromones more quickly, which increases the chance for more ants to chose them. The non-deterministic behavior and the evaporation of pheromones together make it possible for ants to adapt to new situations.

3.2.2. Model

To model the behavior of real ants finding the shortest path Dorigo and Stützle [15] suggest using a simple graph model on which the ants can traverse edges from one node to the next. In the case of the double bridge experiment three nodes (Start, Food, Way) and three edges connecting the nodes are needed to adequately model the situation (see Figure 3). Another simplification for the model is to discretize time

into times-steps $t = 1, 2, 3, ...$ Now $p_{is}(t)$ describes the probability that an ant at

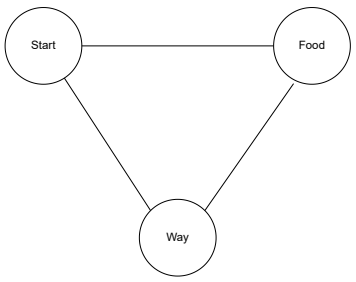

Figure 3.: Model of the Double Bridge Experiment as a graph. The long branch consists of two edges (Start, Way), (Way, Food)

time t at location i chooses the short path s and $p_{is}(t)$ the long path l. To define these probabilities, the pheromone level φ_{ia} an ant encounters at node $i \in (Start, Food)$ on branch $a \in (l, s)$ has to be considered. The long branch l is exactly two times longer than the short branch s and therefore also takes two time-steps to traverse instead of one. This leads to the following equations:

$$p_{is}(t) = \frac{[\varphi_{is}(t)]^\alpha}{[\varphi_{is}(t)]^\alpha + [\varphi_{il}(t)]^\alpha},\ p_{il}(t) = \frac{[\varphi_{il}(t)]^\alpha}{[\varphi_{is}(t)]^\alpha + [\varphi_{il}(t)]^\alpha},$$

where $p_{is}(t) + p_{il}(t) = 1$ and α is a parameter. The update of pheromones on the two branches is described as:

$$\varphi_{is}(t) = \varphi_{is}(t-1) + p_{is}(t-1)m_i(t-1) + p_{js}(t-1)m_j(t-1),$$

$$(i = Start, j = Food; i = Food, j = Start),$$

$$\varphi_{il}(t) = \varphi_{il}(t-1) + p_{il}(t-1)m_i(t-1) + p_{jl}(t-2)m_j(t-2),$$

$$(i = Start, j = Food; i = Food, j = Start),$$

where $m_i(t)$, the number of ants at node i at time t is given by:

$$m_i(t) = p_{js}(t-1)m_j(t-1) + p_{jl}(t-2)m_j(t-2),$$

$$(i = Start, j = Food; i = Food, j = Start),$$

Dorigo and Stützle show in [15], that simulations with $\alpha = 2$ lead to a quick convergence of the ants towards the use of the short branch.
To expand the described system, to find shortest paths in more complex graphs, requires the ants to additionally receive a small memory. This enables them to avoid loops and to remember the entire solution they built. They can then evaluate the built solution to determine the quantity of pheromone to deposit. As mentioned before, the performance can be greatly improved by adding evaporation to the pheromones.

3.2.3. General Ant Colony Optimization

To solve more general minimum cost problems on graphs ACO performs two main steps being the **solution construction** and the **evaluation**.
Firstly, a solution is constructed by an ant beginning at the starting node and incrementally adding neighboring nodes. The decision, which node to add next, is influenced by pheromones deposited in previous iterations. In the very first iteration the pheromone is uniformly distributed along the paths and the decision rule is therefore random. Additionally, a heuristic function η can be used to find better results faster. A common heuristic for short paths is the inverse of the euclidean distance to the destination. Using a heuristic is a greedy approach and a balance between learning via pheromones and preset knowledge via heuristics is crucial. This procedure is repeated until the destination node is reached. Note, in case of grid planning a solution requires all nodes to be added. The order of how the nodes are added and over which edges is stored in memory. The probability of an ant choosing path i considering the heuristic can be formalized in the following way:

$$p_i = \frac{\varphi_i^\alpha \eta_i^\beta}{\sum_{j \in w} \varphi_j^\alpha \eta_j^\beta},$$

where w is the set of all paths reachable from the current position and $i, j \in w$. φ_i is the amount of pheromone on path i and η_i is the heuristic value on path i. α and β are parameters to determine the relation of pheromones and heuristics in the decision of the ant. Furthermore, it holds $\sum_{k \in w} p_k = 1$.
Secondly, the build solution is evaluated and its cost is calculated. Now the amount of pheromone this particular solution should receive is set according to the cost. This could for example be the inverse of the solutions cost. Thereby, solutions with low cost receive more pheromones and vice versa. In the next iteration, the pheromone deposited from the previous solution influences the way in which new nodes are added. Pheromone evaporation can help to reduce the weight that older solutions have on the pheromone level where less information about the problem existed. Generally, more evaporation leads to more exploration of the search space. Evaporation also helps to bound the maximal amount of pheromone achievable. Update and evaporation can be formulated in the following way:

$$\varphi_i = (1 - p)\varphi_i + p\Delta\varphi_i$$

$$\text{with } \Delta\varphi_i = \begin{cases} 1 & \text{if i is part of solution} \\ 0 & \text{else} \end{cases}$$

where p is the evaporation factor and φ_i is the amount of pheromone deposited on path i. A detailed description of the presented ACO algorithm $APMV$ will be presented in section 4.2.

3.3. ACO In Grid Planning

To bring ACO and grid planning together, one needs to formulate the grid planning problem in a way so it can be solved by an ACO algorithm. A suitable way to do this, is to use graphs as a model of the electric grid. Intuitively, the buses of the grid can be seen as nodes (sources, loads are transformers are directly coupled to the buses),

whereas the transmission lines connecting the buses are the edges. Let $G = (B, E)$ be an undirected graph, where the nodes B are the buses $b_1, b_2, ...$ of the grid and the edges E resemble the transmission lines $t_1, t_2, ...$ The goal is to build a graph, where all nodes are connected at the lowest cost possible. Additional constraints like the topology of the graph also have to be considered. As long as the graph is well defined the ACO algorithm can operate in a similar way as previously shown. The node which represents the transformer connecting the grid to the overlying HV level is used as a starting point. It serves as a source to the other buses electricity demand. At the start, none of the nodes are connected to each other and incrementally edges (transmission lines) are added to the graph until all nodes are connected through a path to the starting node (transformer). Afterwards, the cost of such a solution is calculated and the pheromone update is performed on the used paths accordingly.

3.3.1. Delaunay Triangulation

Theoretically, every network station could be connected with all other stations, which is infeasible in practice due to economic reasons. But also for computing, a reduction of the number of potential connections between network stations reduces the algorithms runtime exponentially (reduction of the search space). The Delaunay triangulation is a suitable method to reduce the number of potential lines between stations. It divides a surface with points into triangles by maximizing the minimum of all the angles of the triangles in the triangulation. An example of this can be seen in Figure 4, where the transformer is marked with a square. Research shows, that for the related TSP only 1% of the edges belong to the optimal solution and are not part of the triangulation [16]. The exponential reduction of the number of connections between N buses can be shown via the following equations:

$$e_{total} = \frac{1}{2}N(N-1) \text{ versus } e_{tri} = 3N - 3 - N_h$$

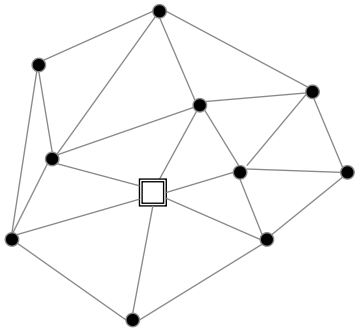

Figure 4.: Delaunay Triangulation of a set of buses and a transformer.

where e_{total} is the number of all possible connections and e_{tri} is the number of possible connections after applying the triangulation. N_h is the number of nodes being part of the convex hull.

3.3.2. Ant Graph

The triangulation can additionally be used to create a new graph called *ant graph* which simplifies the mathematical formulation of the ring network planning problem [3]. To build this graph, the incenter of the triangles and the transformer are taken as nodes (*ant nodes*). Edges are created between triangles, which share a common side (*ant edges*). Since every triangle already constitutes a ring in itself using them to built a solution which satisfies the topological constraints is very helpful. If a neighboring ant node (triangle) is added, the hull of the represented triangles again forms a ring. As long as this ring contains the transformer as a node a valid ring network design is produced. An example is shown in Figure 5, where the unfilled circles represent the *ant nodes* and the dotted lines the *ant path*.

The expansion is started at the transformer (square). In step 1 no *ant nodes* are added to the transformer. The first expansion options are its neighboring *ant nodes*. After the first node is added in step 2, a new expansion option is accessed. The two stations (filled circles) together with the transformer already represent a valid

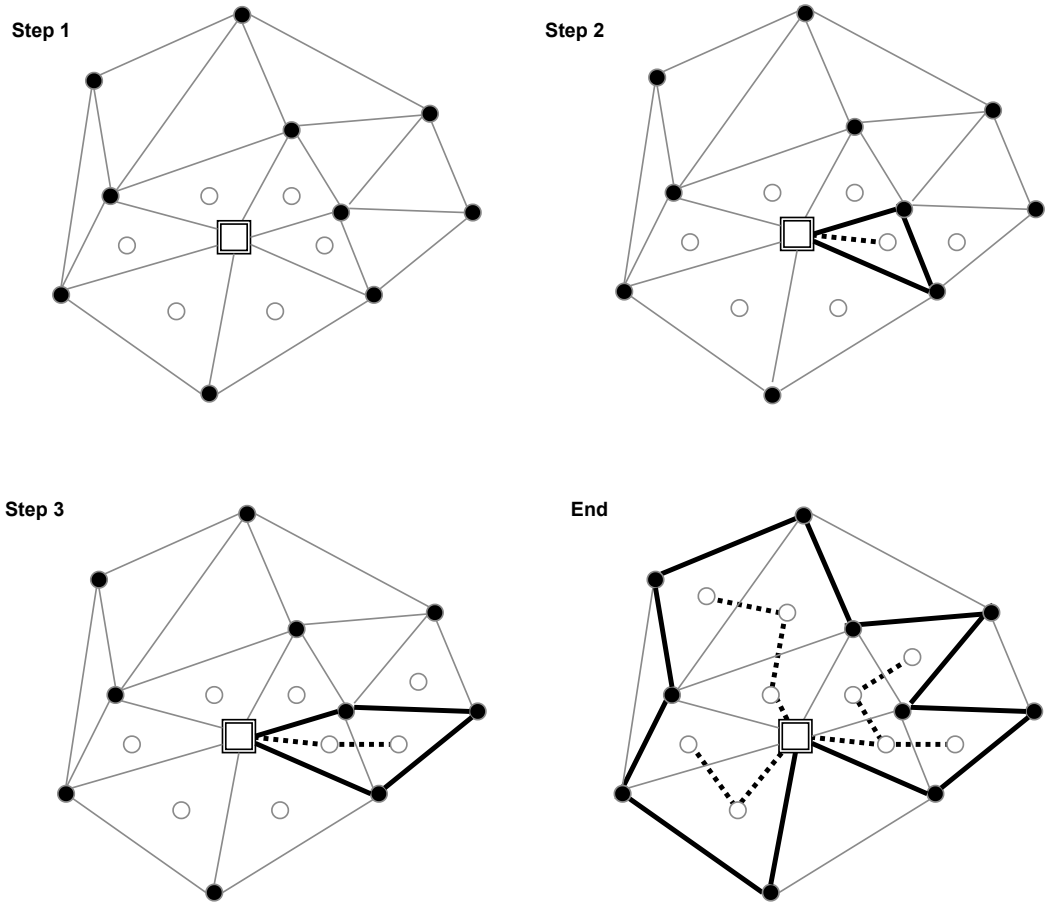

Figure 5.: Building a solution on the ant graph.

ring network structure. In step 3 a second triangle is added, so the hull of the two triangles is the new intermediate ring network solution. This procedure is repeated until all stations are covered by triangles represented by ant nodes.

3.3.3. Switches

When a ring is finished, an open switch is placed at a bus to determine the separation point of the ring. The buses before and after the switch are supplied with power from their respective part of the ring. Only in case of a failure, the switch is closed such that cut off buses are reconnected to the power supply. Thereby, the *n-1 criterion*

of MV grids can be fulfilled. Usually, the switch is placed at a position in the ring, where the load of both parts of the ring are balanced.

4. Approach

First the problem definition is formulated, followed by the presentation of the ACO algorithm *APMV* for solving it.

4.1. Optimization Problem

The goal of the ACO algorithm in grid planning is to connect the loads and generators of the grid in the most cost effective way whilst satisfying the side constraints. The topological side constraint is the *n-1 criterion*, which requires every bus to be reachable via at least one alternative route. This resembles a ring structure in the graph (multiple rings are also possible). As described in 3.3.2, this can be done via finding the spanning tree of the *ant graph* which results in the lowest cost of its hull. The process of building the spanning tree can already stopped, if all stations of the network are covered. There already exist several algorithms, like Boruvkas algorithm, which can find a minimum spanning tree in polynomial time ($O(mlogn)$, where m is the number of edges and n the number of vertices). Unfortunately, the here formulated problem differs from the simple minimum spanning tree problem because not the cost of the *ant edges* but the cost of the hull of the spanning tree has to be minimized. Trying to deduce the cost of the *ant edges* from the cost of the hull is impossible since the edge weight depends on the node from which the edge is expanded. This can be illustrated in Figure 6.
The left triangle has a cost of five, the right triangle a cost of three and their hulls

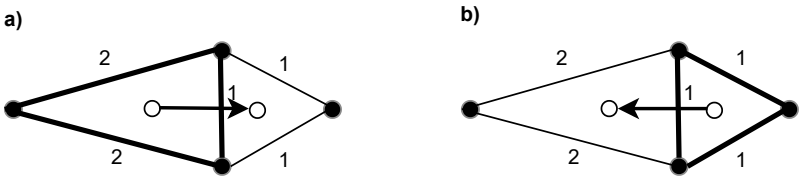

Figure 6.: The cost of an ant edge depends on the node from which it is getting expanded from.

when combined a cost of six. If the edge starts at the incenter of the left triangle and expands the right triangle additional cost of one are added to the hull ($5 + 3 - 2$, since the shared side of the triangles needs to be subtracted). Therefore, one could assign cost of one to the edge. But in the reverse case, the left triangle adds three to the cost of the hull ($3 + 5 - 2$).

Additionally for the problem formulation, two electric side constraints need to be fulfilled. The maximum capacity of the power flow on a transmission line must never be exceeded. Similarly, the voltage violation limits of buses in the network must be complied with. In the case of MV grids in Germany according to DIN EN 50160, the voltage deviation at any bus at any time must not exceed a limit of $\pm$ 5%. The power that flows through a line segment shall never be more than 50% of the lines maximum capacity. The voltages and power flows in the network can be calculated via a power flow analysis (PFA).

4.2. Ant-Power-Medium-Voltage

The algorithm *APMV* was developed to solve the optimization problem formulated in the previous section. It is an advancement of the algorithm presented by Zeller in [13] and uses concepts presented by Dorigo et. al. and Rotering.

4.2.1. Input and Parameters

The algorithm was specifically designed for the medium voltage level. If the input is a mixed grid and consists of multiple voltage levels, the medium voltage part needs to be extracted first (see 5.1.3). As a data format the algorithm accepts a *PyPSA*-network (see 5.1.1) as an input file.
Important Parameters to set are the following:

- **lineType**: Specifies the type of the line which should be build. This determines the lines resistance, reactance and capacity of its maximal power flow.
- **antsPerColony**: How many ants are building a solution in every iteration.
- $q0$: Probability that the ant chooses the best option (according to pheromone level + heuristic).
- α: Parameter to determine the influence of the pheromones in the expansion decision.
- β: Parameter to determine the influence of the heuristic value in the expansion decision.
- ρ: Parameter for the global pheromone update. Determines how much pheromone is deposited to reward the best solution.
- ξ: For the local pheromone update. Determines how much pheromone is removed from already used paths to evoke more exploration.
- **iterations**: Number of iterations of the algorithm.
- **maximalRingsize**: Maximal number of buses allowed per ring.

4.2.2. APMV Implementation

In algorithm 1 the high-level implementation of *APMV* is shown. At the very beginning, the input grid is prepared to be compatible with the algorithm in a preprocessing step. Afterwards the initial pheromone is deposited. As already explained in Chapter 3.2, every ant in the algorithm builds one solution per iteration. Out of these

solutions the best one is kept and rewarded with pheromones which influences the solution building process of the next iteration (*globalPheromoneUpdate*()). After a postprocessing step, the overall best solution and its cost is returned. φ_0 is the pheromone distribution at the start and N is the number of stations in the network.

Algorithm 1 High-level implementation of *APMV*

```
1: preprocessing(grid)                          ▷ extraction of MV grid and triangulation
2: φ_0 ← initializePheromones()                 ▷ 1/N, with N = number of stations
3: lowestCostSoFar ← ∞
4: foreach i ∈ iterations do
5:     solutionsPerIteration ← ∅
6:     foreach ant ∈ antsPerColony do
7:         solution ← createSolution(φ_i)
8:         solutionsPerIteration ← solutionsPerIteration ∪ solution
9:     end for
10:    bestSolutionIter ← argmin(cost(), solutionsPerIteration)
11:    if cost(bestSolutionIter) < lowestCostSoFar then
12:        lowestCostSoFar ← cost(bestSolutionIter)
13:        bestSolution ← bestSolutionIter
14:    end if
15:    φ_i ← globalPheromoneUpdate(φ_{i−1}, bestSolution)
16: end for
17: postprocessing(bestSolution)                ▷ export back into PyPSA format
18: return bestSolution, lowestCostSoFar
```

Global Pheromone Update

The function *globalPheromoneUpdate()* is performed once after each iteration and deposits additional pheromones on the nodes expanded by the best solution found so far. Over time this enables the algorithms to learn from previous iterations and shows which nodes generally lead to good or bad solutions. The formula for the global pheromone update on all visited nodes i of the best solution is the following:

$$\varphi_i = (1 - \rho)\varphi_i + \rho \frac{c_{est}}{c_{best}},$$

where $\rho \in (0, 1)$, c_{best} are the cost of the best solution found so far and c_{est} are the estimated cost of the solution. The reciprocal of the c_{best} is is used such that low cost are rewarded by high amounts of pheromone. It is normalized by the c_{est} to better reflect the quality of the solution. As long as $\frac{c_{est}}{c_{best}} > \varphi_i$ the pheromone increases. Otherwise, the pheromone level can also decrease.

4.2.3. Solution Creation of Ants

In algorithm 2 the schematic implementation of the *createSolution*() function is shown. The expanded nodes are ant nodes and the *solution*() method checks whether the hull of the set expandedNodes covers all stations. The *expand*() method finds the next node which should be added and depends on the pheromones φ, the heuristic η and the neighborhood *neighbor*() of the already expanded nodes. In the $localPheromoneUpdate()$, the pheromone gets decreased on the already used path to increase exploration and reduce the chance of convergence towards a local minimum. When *solution*(expandedNodes) is true, a solution is found and is returned with the updated pheromone level φ.

Algorithm 2 Implementation of createSolution(φ)

```
1: expandedNodes ← ∅
2: while solution(expandedNodes) = false do
3:     newNode ← expand(φ, η, neighbor(expandedNodes)
4:     expandedNodes ← expandedNodes ∪ expand(φ, η, neighbor(expandedNodes))
5:     φ ← localPheromoneUpdate(newNode, φ)
6: end while
7: return expandedNodes, φ
```

Local Pheromone Update

The function *localPheromoneUpdate()* is executed every time the ant expands a new node and generally reduces the amount of pheromone on that node. This should

increase the chance of other ants to use different paths instead and therefore increase exploration. The pheromone level of node i gets updated via the following rule:

$$\varphi_i = (1 - \xi)\varphi_i + \xi\varphi_{init},$$

where the parameter ξ determines, how much the pheromone level should revert back to the initial pheromone level φ_{init}.

Heuristic

The purpose of the heuristic η is to provide additional guidance for the ant when making the decision which node to expand next. Especially in earlier iterations when the algorithm has not learned much yet (via pheromones) the heuristic can lead to better quality solutions. The overall cost of a built solution is roughly proportional to the length of the rings. The aim is to estimate the additional cost of expanding a node, so the ant can chose the one with the lowest cost. When adding a node (which represents a triangle) the length of the ring increases by the perimeter of the triangle minus two times the shared side (see Figure 7 a)).

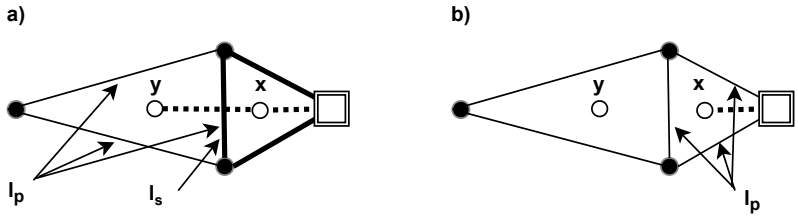

Figure 7.: Calculating the heuristic value.

The reciprocal of the length a triangle (ant node) adds to the ring is used as a heuristic.

$$\eta_{x,y} = \frac{1}{l_p - 2l_s}$$

As explained in 4.1 this heuristic not only depends on the expanded node y, but also on the node from which it gets expanded x. This leads to high heuristic values for triangles with short sides and low heuristic values for triangles with long sides. For the first triangle that gets expanded from the transformer node, the heuristic value is just the sum of the length of all sides, since there is no shared side (see Figure 7) b). The problem with this (greedy) heuristic is that it does only minimizes the next expansion step and not the length of the ring at the end. Additionally, it does not account for the cost of voltage violations on buses.

Expansion rule

As shown in algorithm 2, the *expand*() method depends on the heuristic η, the pheromones φ and is limited to the neighbors of the already expanded ant nodes. As suggested by ACS [11], the next node $s \in neighbor(x)$, with x being the already expanded nodes and $s_i \in neighbor(x)$, is chosen according to the following rule:

$$p_s = \begin{cases} argmax_{s_i}(\varphi_{s_i}^{\alpha} \eta_{s_i}^{\beta}) & \text{, if q} \leq q_0 \\ \frac{\varphi_s^{\alpha} \eta_s^{\beta}}{\sum_{s_i} \varphi_{s_i}^{\alpha} \eta_{s_i}^{\beta}} & \text{, if q} > q_0 \end{cases}$$

where $q \in [0, 1]$ is a uniformly distributed random variable. If the value of q is greater than the threshold q_0 the algorithm choses the best known path (exploitation) and if q is lower than q_0 the algorithm choses randomly (exploration).

4.2.4. Switches

APMV places the switches exactly in the middle of the ring (regarding the number of stations). This is done regardless of load or generation of its buses. Most of the time, this leads to fairly balanced loads on both parts of the ring. In future work

however, switches could be placed based on the load and generation distribution inside the ring to ensure a good balance.

4.2.5. Cost function

The cost function plays a crucial part in the algorithm since it determines the pheromone distribution of the next iteration and therefore how the algorithm improves. The costs consist of two parts, the cost of building lines $b(x)$ and a penalty term for violations of electrical constraints $e(x)$.

$$cost(x) = b(d(x), c(x)) + e(x),$$

where x is the network of the solution, $d(x)$ are the digging cost and $c(x)$ are the cable cost. The building cost and the penalty term will now be discussed in further detail.

Building cost

Concerning the cost of building lines, there are the acquisition costs of new transmission cables and the digging costs of laying the cables into the ground. As an estimation for standard MV lines are taken cable cost of 130K Euro/km and digging cost of 100K Euro/km. These values of course fluctuate over time and have to be adapted for the specific situation. If the algorithm suggests building a line between two buses, where a line already exists, the costs are set to zero (assuming they can still be used in the future). For new transmission lines, the algorithms tries to predict the length and where the cable is laid along via the street network. Usually, cables are laid next to streets for better access and less invasive construction works. This is taken into account by using a tool developed by John [17], which uses data from open street map to find the distance and the path between two network stations alongside streets. John first creates a graph from open street map in a given area, connects

the network stations to it and then calculates the shortest path between the stations using Dijkstra. If the distance between two stations cannot be estimated via open street map, the straight distance calculated via the haversine formula multiplied by 1.2 is used.

Penalty term

The second part of the costs is a penalty term accounting for voltage violations of buses and loading violations of lines of the target grid. Therefore, a PFA is conducted, which calculates the voltages at all buses and the power flowing in each line of the network. If values exceed their limits penalties are added to the cost. These penalty costs are orders of magnitude greater in comparison to the building costs. Thereby, the ants get rewarded much more by finding solutions without load flow violations. With enough iterations this practically guarantees that a convergence towards a solution has either no load flow violations at all or it is impossible to find a solution without it.

4.2.6. The Back-and-Forth Case

As Zeller already points out in [13], there exist cases in which the standard process of building the hull of the *ant graph* as described by Rotering does not work. Such a case is shown in Figure 8. If the expansion of ant nodes proceeds as shown in case b), one station is not contained in the hull. The problem here is that it is possible to expand a triangle where all its three stations are already covered by the hull at that point in time (t5 in c)). This is necessary though because the triangle t6 is only reachable via t5. Zeller's approach is to allow the expansion of the crucial triangle t5 anyways, but to do the routing in a way so it maintains the ring structure and it does not lose the connection to the node in the middle (Zeller called this a *back-and-forth* case). Unfortunately, his solution only worked in this specific case and did not cover

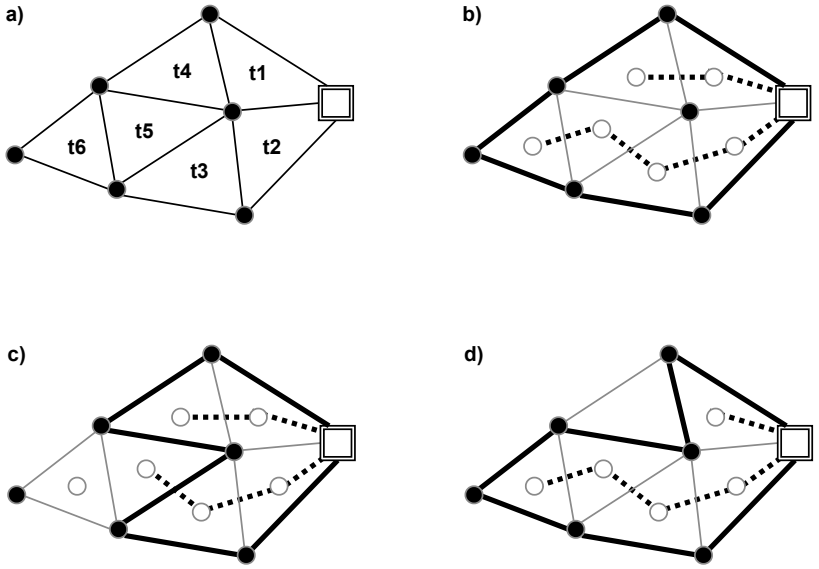

Figure 8.: Problematic case of building a ring via the hull of the ant graph.

further problems caused by the same issue. *Back-and-forth* cases can be chained indefinitely and Zeller only solved them until a certain depth. Another possible attempt of solving this issue would be to create two parallel lines from the hull to the unconnected station in the middle. This would maintain the ring structure but also connect the outer station twice. Therefore, *APMV* prohibits the expansion to triangles where all their three stations are already covered. If such a case occurs, the ant is getting restarted and a different solution is constructed (d). Depending on how often the ant needs to get restarted the performance of the algorithm reduces. During the creation of this work it was not clear how often such cases would appear in real networks and if it would be worth to develop an exact mechanism, which covers all problematic cases. For the real network example evaluated in this work however, a run of 1000 iterations and 15 ants per iteration only required 40 ants to be restarted, which does not represent a major issue ($\frac{40}{15000} = 0.00226\%$). I leave for future work to find a method for an exact solutions to this problem.

4.2.7. Multiple Rings

To increase security it is common to limit the amounts of stations per ring to a certain maximum. Therefore, it is possible that multiple rings are required to cover all stations in the network. Another reason might be that rings connecting to different HV/MV transformers overlap in certain parts. Although it is possible to have multiple lines on the same path, the buses must only be connected to exactly one ring. To avoid the connection of buses to multiple rings it would be possible to check during the solution creation whether a bus is already covered by a different ring and to forbid its expansion. This however could lead to many dead ends, since the algorithm is limited to the expansion to neighboring nodes created by the triangulation. Therefore, *APMV* allows the rings to expand to every neighbor independent of their potential connection to a different ring and to resolve the issue in a postprocessing step. Figure

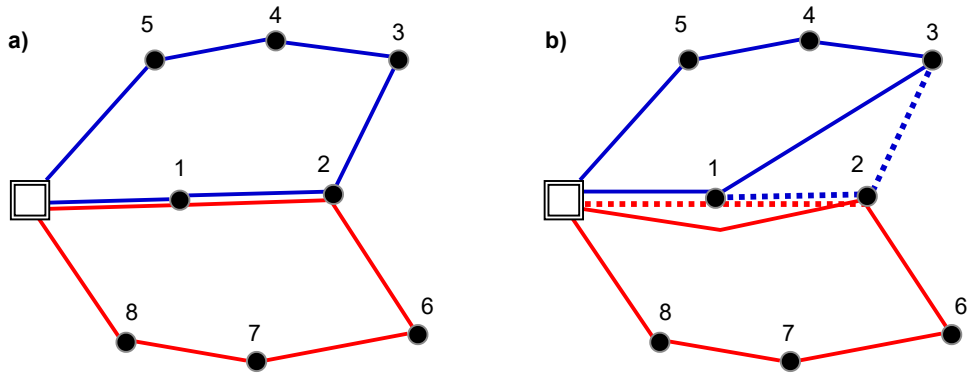

Figure 9.: Handling of overlapping rings.

9 a) shows a case where two rings (blue and red) are partly overlapping at buses 1 and 2. To resolve the issue bus 1 gets chosen to be connected to the blue ring and bus 2 to the red ring. The connections of buses to lines are depicted by the continues lines and how the cables are put into the ground is illustrated by the dotted line. The lines basically stay where they should be according to the triangulation but the buses are just connected to maximal one line. *APMV* determines the allocation of buses to rings randomly. In future work this could be based on balancing the load

and generation of rings.

4.2.8. Ring Size and Triangulation

Limiting the number of stations can be done easily during the expansion process of the ant. When an ant expands a new triangle during the solution building process, the number of stations of the ring also increases by one (neighboring triangles already share two corners). This applies to all triangles except the very first one expanded from the HV/MV transformer, where two stations are added. The number of stations per ring is therefore the number of expanded triangles incremented by one. When the ant reaches the maximum number of triangles it is allowed to expand, it stops with the expansion and starts constructing a new ring starting from the HV/MV trafo. Downwards, the maximal number of stations per ring is limited depending on the structure of the grid.
When the maximal number of stations per ring is limited, another complication in conjunction with the triangulation can arise. Namely, the number of neighboring triangles of the transformer is limiting the number of rings which can be built. A

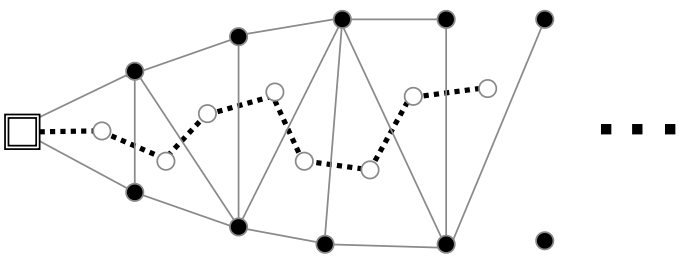

Figure 10.: Limitations of triangulation and limited ring size.

worst case scenario is depicted in Figure 10. There can exist an arbitrary number of stations in the network but through the triangulation only one neighboring triangle to the transformer is built. After the first ring reaches its maximal number of stations, there is no triangle left for another ring to start its construction. To counteract this bottleneck, the expansion rule described in 4.2.6 is relaxed for neighboring

triangles of the transformer. This means that it is allowed to expand a triangle even though all three of its stations are already covered. In general this technique helps to find more solutions, although in the shown example it also would not work, since there only exist one available expansion option for each triangle. Another possible fix, is to increase the maximal number of allowed stations per ring. Similar to 4.2.6, *APMV* just tries to restart the solution building process in the hope to find a different expansion order, which does not result in a restart. For the real network evaluated in this work, the number of restarts out of 1500 attempts in relation to the maximal ring size is depicted in Figure 11. For a maximal ring size of 7, every solution building attempt has to be restarted, so no solution could be found. Manual verification confirms that it is impossible to create a solution with only 7 stations per ring. In practice however manual verification is too time consuming, therefore sufficiently many iterations and ants per iteration have to be chosen. For 8 or more rings valid solutions are found by the algorithm. When the ring size is unlimited, almost all attempts lead to valid solutions. For critical ring sizes though runtime can be compromised considerably.

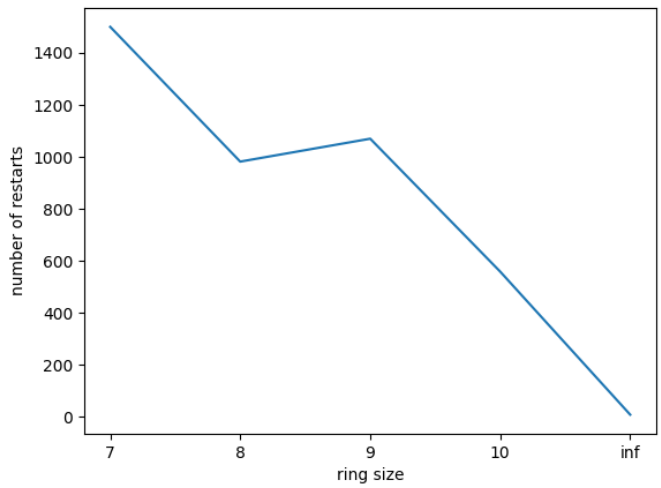

Figure 11.: Number of restarts for different maximal ring sizes.

The approach of $APMV$ is to prevent the expansion of critical triangles which could lead to faulty network structures already during the solution building process. The reason not to fix an invalid solution in a postprocessing step is that it seems more practical. Without preventing the expansion of certain triangles, it is impossible to trace back and classify the exact reasons for a failed solution building attempt in hindsight. And without this classification of errors a corresponding "reparation" is very difficult. For future work, it might be an interesting task to find a postprocessing procedure which fixes invalid solutions created with an unrestricted expand method. This would prevent the need for restarting the solution building process and could therefore speedup the algorithms runtime.

4.2.9. Run Time

The time complexity of the algorithm mainly depends on the number of constructed solutions $n_{col}n_{ants}n_{iter}$ and the size of the network, where n_{col} is the number of colonies, n_{ants} is the number of ants and n_{iter} is the number of iterations. Every constructed solution performs a PFA which has a time complexity of $O(B^3)$ in the worst-case, with B being the number of buses. Additionally, every solution must expand nodes until all nodes are covered. This could take up to B expansion steps. In the decision process the ant needs to consider up to B possibilities which bus to expand next. This leads to $O(B^2)$ steps in the worst-case. In total this sums up to a runtime of $O(n_{col}n_{ants}n_{iter}(B^3 + B^2))$. However, depending on the structure of the grid and the input parameters an arbitrary number of restarts of the ant could slow down the algorithm indefinitely.

5. Experiments

5.1. Settings

5.1.1. Libraries

A key role for the implementation plays *PyPSA* (Python for Power System Analysis) [18], which is a toolbox to simulate and optimize power systems. The components of its network container are very closely related to the network components presented in section 3.1.1. *PyPSA's* buses are the fundamental nodes to which all loads, generators, lines and transformers attach. In accord with Kirchhoff's current law everything feeding in out of *PyPSA's* buses is balanced out, to enforce energy conservation. The data is stored in *pandas* DataFrames to enable efficient calculations. *PyPSA* is also able to perform a PFA on the network via the *Newton-Raphson* method. Among other uses, this is especially important for the calculation of the penalty term. *Networkx* is used by the algorithm to store and modify the graph which represents the electric grid. For plotting, the library *Matplotlib* is used.

5.1.2. Example Grid

For testing and evaluating the algorithm the grid of a village in rural Germany is used (exact name and location is omitted due to data protection reasons). It is a mixed grid which consists of a medium and a low voltage level. It connects to the higher voltage level from the south. It consists of 1857 buses, 1851 lines, 16 MV/LV

transformer stations, which connect the medium voltage level to the underlying low voltage level, and 1 HV/MV transformer which connects the whole network to the upstream high voltage grid.

Although the network contains only 17 transformers, the number of possible rings, which connect the transformers is very large ($\frac{17!}{2} \approx 1.77*10^{14}$ [number of permutations divided by two, since the exact reverse order of stations is considered the same ring]). That is why even on this example a naive brute force approach would not be feasible.

Figure 12.: The grid used for testing. (internally developed tool used for visualization)

5.1.3. MV Grid Extraction

The algorithm was specifically designed for the medium voltage level and therefore the input grid needs to fulfill the following requirements:

- The input grid should contain only medium voltage buses (20 kV), which resemble the MV/LV transformers.
- All buses must have coordinates (for the triangulation).
- The buses loads are an aggregation of the loads of the underlying LV grid.
- The input grid only contains MV transmission lines (20 kV)
- The lines always directly connect two MV buses.
- There has to exist at least one HV/MV transformer (which also functions as a Slack bus) from where the algorithm can start.

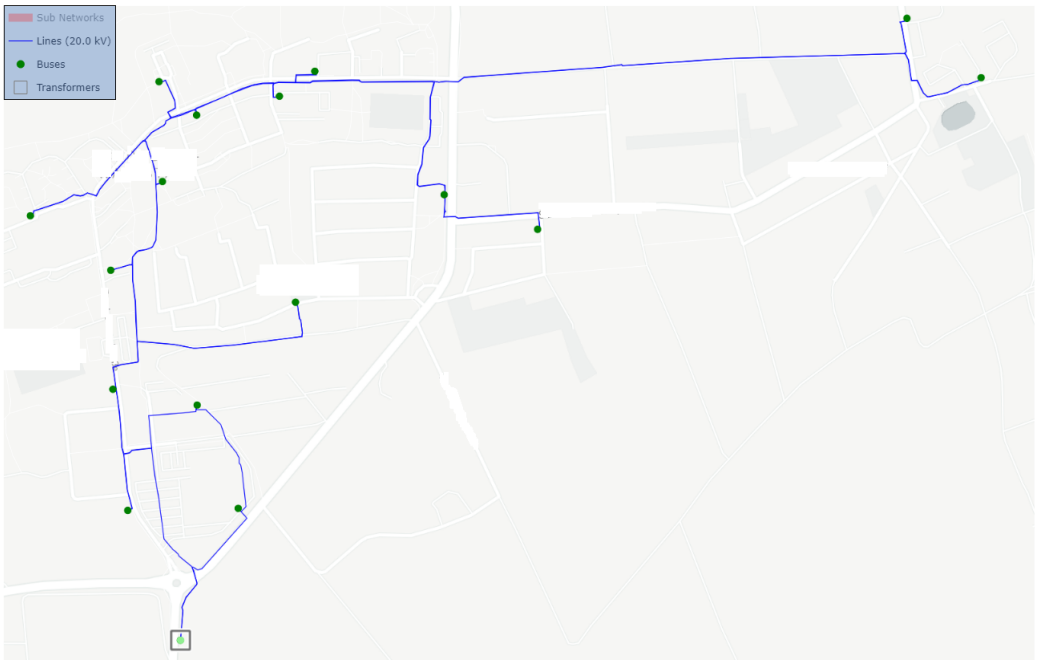

Figure 13.: Middle voltage part of the example grid.

In a preprocessing step the MV part of the original grid is extracted (see Figure 13), to bring the input of the algorithm in the required form. After the extraction only the MV buses of the transformers are kept (in *PyPSA* transformers connect two buses of different voltages). The task of the algorithm is to connect the buses via transmission lines in the most cost effective way, while satisfying the given side constraints. The blue lines are MV cables which already exist in the network and which can be used by the algorithm at zero cost. The green dots resemble the buses.

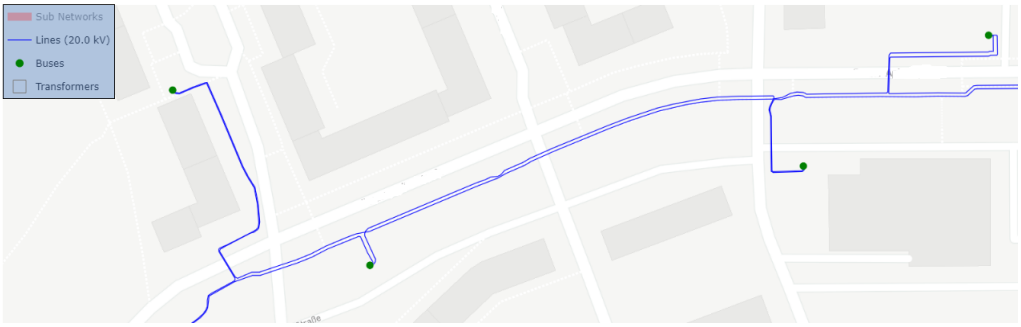

Figure 14.: Parallel lines in the ring of the MV grid.

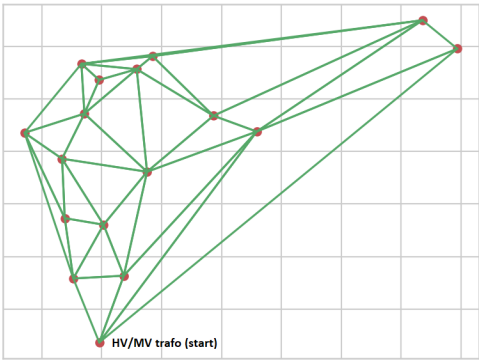

Figure 15.: Triangulation.

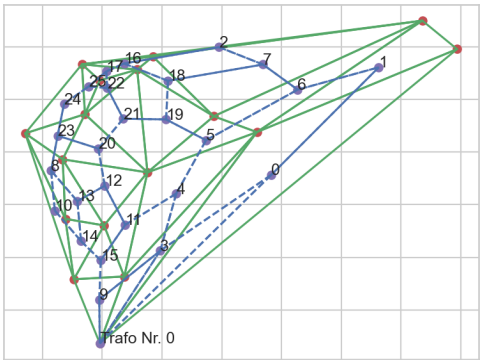

Figure 16.: Ant graph.

At first glance, the topology of the extracted MV grid does not look like it possesses the required ring structure. Only when zoomed in it is possible to see, that through parallel lines, the ring structure is maintained (see Figure 14). It is debatable however, whether is it useful to put two lines in parallel to achieve adequate security. More to this in the discussion section 5.2.2.

In Figure 15 the triangulation of the buses of the MV grid is shown in a schematic view. From the triangulation, the ant graph is built, which can be seen in Figure 16. The blue ant nodes are the incenter of the triangles. Neighboring ant nodes are connected via an edge (dotted lines do not have a specific meaning in this context).

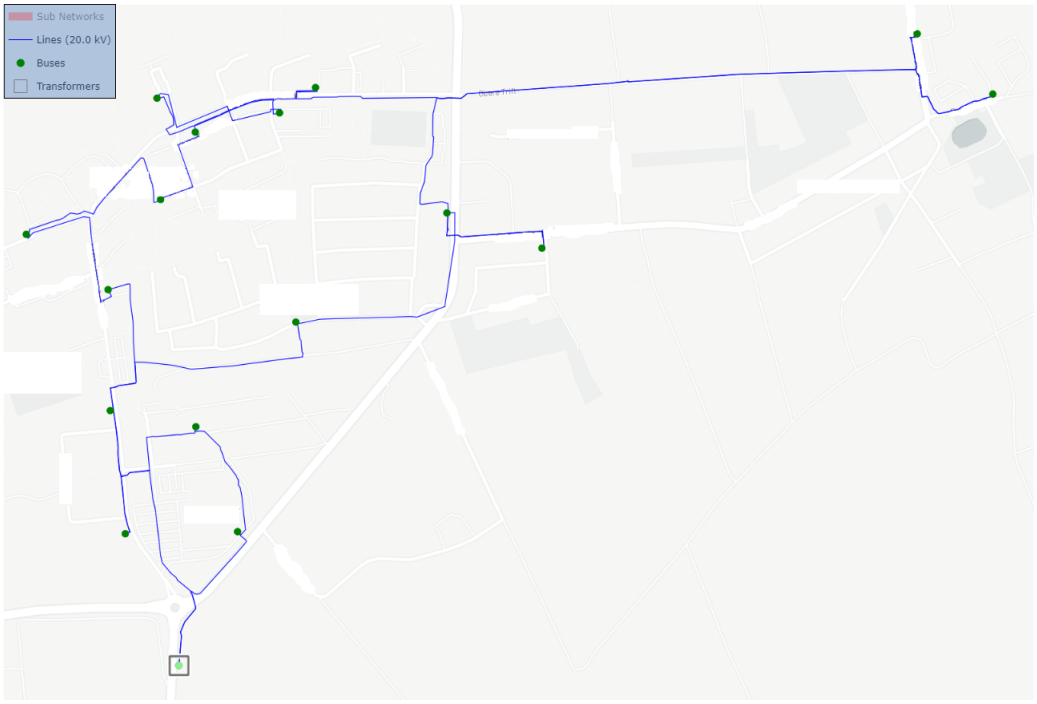

Figure 17.: Result network of best found MV grid.

5.2. Results

The best result found by *APMV* with default parameters is shown in Figure 17. Most of the lines which are used in the input grid are also part of the result grid, since they can be reused at zero cost. Four lines are new and added by the algorithm alongside streets. The penalty term for electric constraint violation can be kept to zero.

The associated costs are depicted in Figure 18. The minimal cost found by the algorithm declines rapidly in the first iterations and then plateaus out after a small drop at around iteration 150. This last decrease in cost was not captured by other runs, since they were capped at 100 iterations. The step-wise decrease in cost is due to a new and cheaper routing being found (drop) after which the algorithm produces only equal or worse solutions (plateau). Important to notice, is that the minimal found cost of ca. 309K Euro does not reflect the total cost of the network,

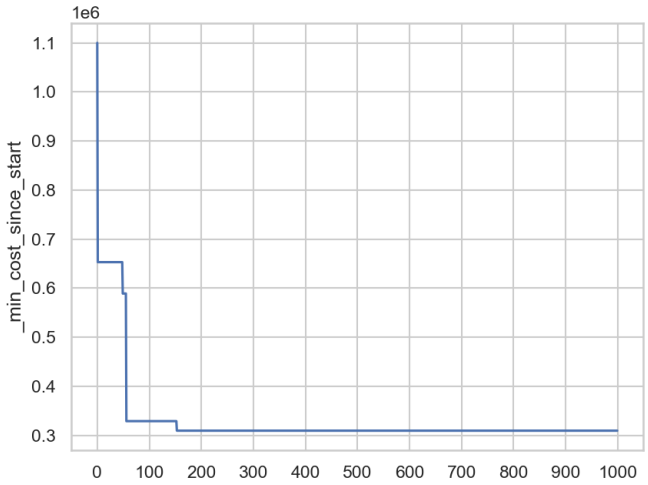

Figure 18.: Minimal cost found since the start.

but only the lines additionally built by the algorithm. Already existing lines do not appear in the cost function since their cost is zero. To calculate the total cost of the solution including the existing lines, all lines are considered to be newly built by the algorithm in a post processing step. This leads to a total cost of 1.95M Euro for the lines of the entire MV grid. These costs can now be compared to the same calculation done with the extracted MV grid of the input. This results in costs of 2.17M Euros. The found solution of the algorithm is therefore only 89% of the cost of the input grid. The real building cost of the grid example however depends on many more factors, which could not all be considered.

To better see how the algorithm connects the buses a schematic view of the solution is given in Figure 19. Here, the rings structure is clearly visible.

5.2.1. Parameterstudy

As listed in section 4.2.1, the algorithm has fairly many input parameters. Trying out multiple values for each parameter in all possible combinations would therefore be unfeasible. Instead, single parameters are are varied while the remaining parameters

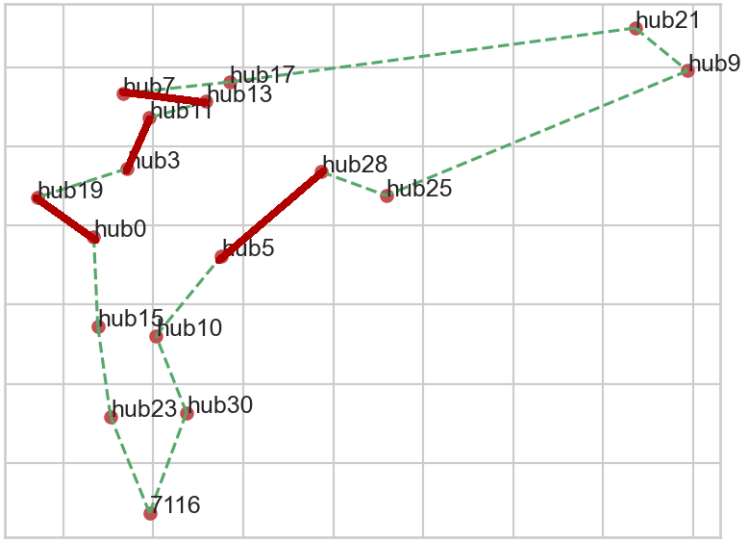

Figure 19.: Schematic view of the best found MV grid. The "hubs" resemble MV/LV trafos which need to be connected to "7116" - the HV/MV trafo. Newly created lines by the algorithm are marked red.

are kept at their default values. When only single parameters are varied, interactions between parameters can unfortunately not be traced anymore.

The parameter study is divided into four parts. First, the parameter for maximum number of stations per ring is studied (maximalRingsize). The second part investigates the parameters that changes the number of constructed solutions (antsPerColony, iterations). The third part examines the parameters controlling the expansion rule ($q0$, α, β) and the last part studies the parameters controlling the pheromone update (ρ, ξ).

Control Maximum Number of Stations per Ring

In the first part the maximal number of buses per ring are limited to a fixed value. Limiting the number of stations per ring can be useful to reduce the loading of the lines due to a smaller number of connected loads. Additionally, it decreases the number of stations being cut off electricity in case of a failure. The smallest number of stations per ring for which the algorithm was able to find a solution is eight.

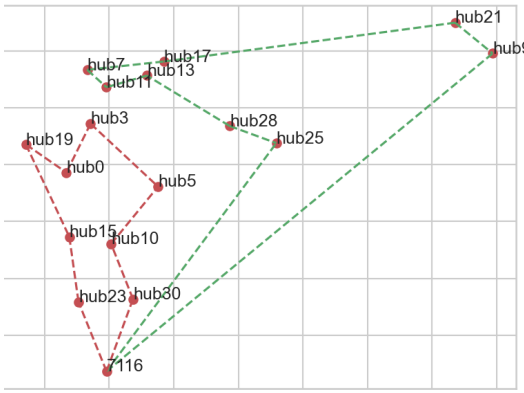

Figure 20.: Max. 8/9 buses per ring.

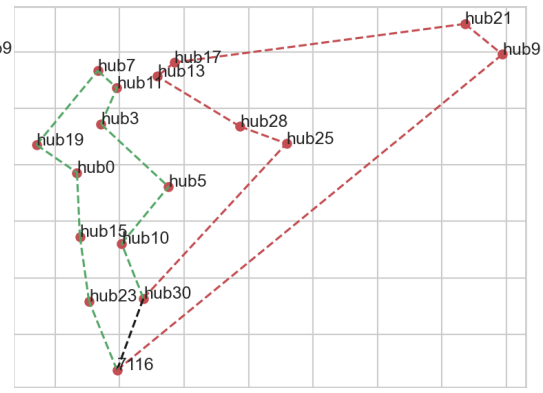

Figure 21.: Max. 10 buses per ring.

For less then eight stations per ring it is impossible to build a solution, due to the problem formulated in section 4.2.8. There exist only three neighboring triangles to the start transformer and each ring occupies at least one of them. If all neighbors of the start transformer are blocked but some stations are still not connected, the ant must perform a restart. In case of maximal seven stations per ring, the ant restarts every solution building attempt out of 1500. A Manual check confirms that it is indeed impossible for the algorithm to find a solution.

Figure 20 and 21 show the rings being build (schematic view), when limiting the maximum number of stations to 8,9 and 10 (see appendix for non schematic view). The predicted cost of building the network as suggested in Figure 20 would amount to 2.53M Euro (117 % when compared to te original network) and 2.52M Euro (116 %) for the network suggested in Figure 21. In general, it makes sense that increasing the number of rings also increase the cost, since additional cables are required to return to the HV/MV transformer. For more than 10 but less than 16 stations the algorithm found the same solution as shown in Figure 21 and for 16 or more stations the algorithm suggested building a single ring, which is the cheapest option for this grid.

Control Number of Constructed Solutions

To perform well, ACO algorithms need to learn from previous iterations and explore a sufficient area of the search space. Even though the heuristic function can lead ants quickly towards reasonable results, the number of constructed solutions plays a major role. *Ant-Colony-System* [11] by Dorigo et. al. presents three parameters to control the number of constructed solutions. The number of iterations, the number of colonies and the number of ants per colony. The ants solution building process is only dependent on the best solution of the previous iteration and not on the solutions built by other ants in the same iteration. Therefore, a second independent colony of ants could create solutions in parallel. *APMV* however is a sequential algorithm, so the number of colonies is always set to one. In future work, the feature of multiple colonies per iteration could be added to speed up the algorithm.
How the minimal found cost change with respect to the number of iterations can already be seen in Figure 18. On this test grid the greatest reduction in cost is already achieved after ca. 50 iterations. Around iteration 150 another small decrease in cost can be observed. Due to limited computational resources though, the default number of iterations is set to 100. An automated stop of searching for a further decrease in cost after a certain amount of iterations could be useful.

The number of ants per iteration is the third parameter to control the number of constructed solutions. Its effect is shown in Figure 22, which depicts the minimum cost found so far, for different number of ants per iteration. The cost for 20 and 25 ants per iteration is congruent with the cost for 30 ants per iteration (this might be caused by the usage of the same seed in the algorithms random functions). For 20, 25 and 30 ants the algorithm yields the best results. Using less ants per iteration might lead to less exploration and therefore a convergence towards a local minimum. In the future it would be interesting to increase the number of ants even further until the performance would decline again.

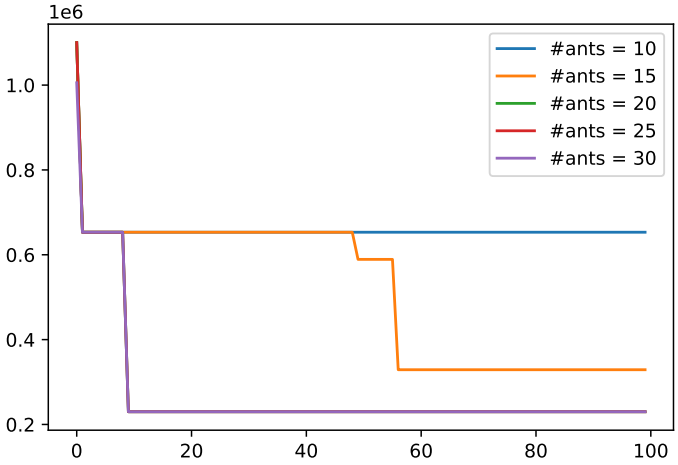

Figure 22.: Minimal cost found since the start for different number of ants per iteration.

Control Expansion Rule

The parameters which control the expansion rule are $q0$, α and β. They guide the ant in its decision which node to expand next. The parameter $q0$ determines to which extend the ant is influenced by knowledge of the best solution of previous iterations. It modulates the trade-off between exploitation and exploration. Figure 23 shows the minimal cost found at a certain iteration with respect to different values of $q0$. For $q0 = 1$ the ant builds a solution and is forced afterwards to always expand the exact same components since no exploration is allowed. That is why the ant can never explore a better solution and therefore the costs can never decrease. The best solution is found for $q0 = 0.75$, since it enables the ant to explore more of the search space. Similar to the number of ants in the previous subsection it would be interesting to investigate even lower values for $q0$ in the future, until the performance of the algorithm would decline again.

The parameters α and β describe how much the ants decision process of expanding new nodes should be guided by pheromones (α) or the heuristic value (β). The smaller the value, the lower the influence. Figure 24 shows, that the best results are

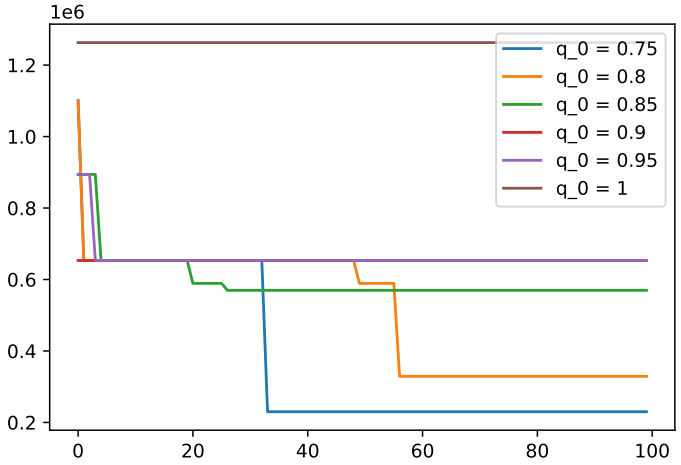

Figure 23.: Minimal cost found since the start for different $q0$.

achieved with $\alpha = 0.2$ and $\beta = 0.8$, where the cheapest solution is already found around iteration 25. For $\alpha = 0.4$ and $\beta = 0.6$ the algorithms also finds the cheapest solution but only in a later iteration, shortly before 60. This means, that giving the heuristic a higher weight than the pheromones, when making the decision which node to expand next, yields better results. It makes sense that especially for lower iterations the algorithm can make better use of guidance via the heuristic, since it has not learned enough from previous iterations yet. It would be interesting to see if the results would get better for a higher weight of pheromones in later iterations.

Control Pheromone Update

The last parameters to study are ρ and ξ, which influence how the pheromones are updated. ρ determines how much pheromone is added during the global pheromone update and ξ determines the amount of pheromone being removed during the local pheromone update. The global pheromone update is performed at the end of an iteration and only applies for the best built solution. Its goal is to reward good solutions such that future iterations can use high pheromone trails as guidance. The

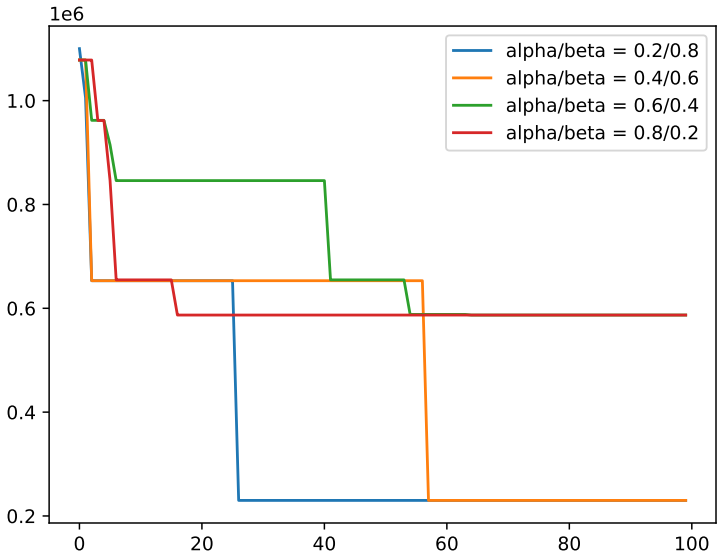

Figure 24.: Minimal cost found since the start for different configurations of α and β.

local pheromone update occurs every time an ant expands a new node and reduces its pheromone level. This promotes the usage of new paths and therefore a greater exploration of the search space. Figure 25 shows that for ρ all values yield fairly good results. For the tested network the best solution can actually be found for $\rho = 0$, which means that the global pheromone update does not change the value of the pheromones according to the best solution. This is surprising since it is a necessary step of the algorithm in order to learn from previous iterations. The reason for the good performance for $\rho = 0$ might be that for higher values of ρ the ant lingers for a longer time in a local optimum (due to a previously found solution) instead of exploring more of the search space. More iterations could help to see whether the cheapest solution would be found (by values of $\rho > 0$) eventually.

Figure 26 depicts the minimum cost found for different values of ξ. Best results are achieved for the higher values $\xi = 0.3$ and $\xi = 0.2$. A higher value of ξ means that the pheromone of used nodes is converging back faster to its base level and therefore

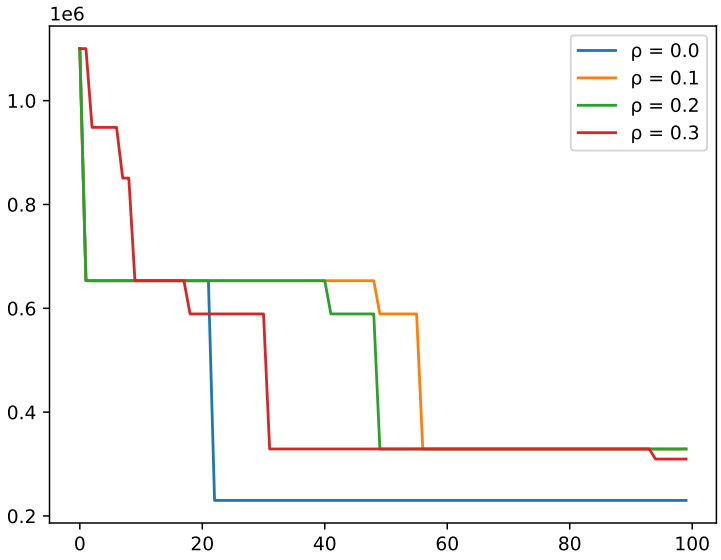

Figure 25.: Minimal cost since start for different values of ρ.

more exploration. This finding fits with previous results for the study of parameter ρ.

5.2.2. Discussion of Triangulation and Parallel Lines

As presented in section 5.1.2, the example grid uses parallel lines to maintain the required ring structure for medium voltage grids. This increases security of electricity supply, since damage to a singular line does not lead to a power outage. A common source of transmission line failure is that cable sleeves, which are used to connect cables, pull water from the surrounding soil. Such failures only affect single lines and can be prevented from leading to power outages by using a secondary line directly next to the first one (parallel line). In case of greater accidents caused by large construction works or natural disasters parallel lines do not necessarily provide additional safety since damage at one cable mostly also affects the secondary one. Here, a ring structure without parallel lines would provide more security. Allowing

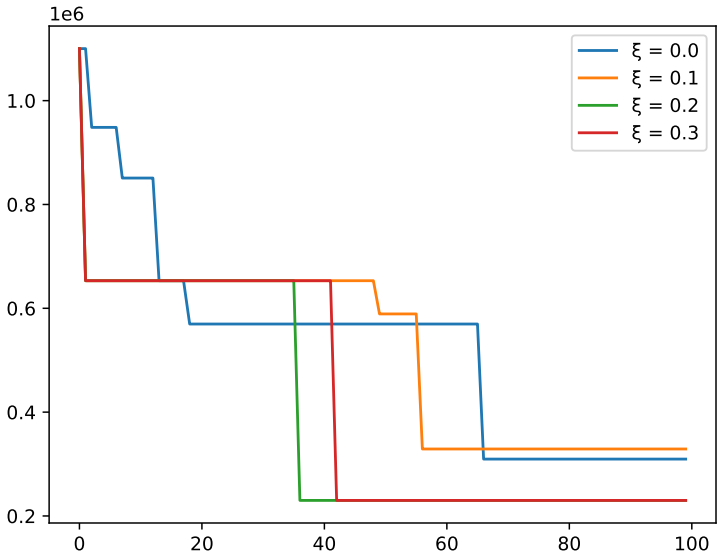

Figure 26.: Minimal cost since start for different values of ξ.

lines to be built in parallel however also reduces cost, since multiple cables could be laid into the same trench.

In the case of *APMV*, the algorithm uses the triangulation and the hull of the ant graph to build rings and therefore parallel lines are generally not provided for. When two buses should be connected by a line (and no line already exists between them), the shortest route alongside roads is used to lay the cable. This coincidentally can result in parallel lines as can be seen in in the upper right part of Figure 17, but does not have to. This is also the reason why the algorithm does not simply return the input grid although existing lines can be reused at zero cost. Especially, if the input network has fewer already existing lines, the chance of parallel lines decreases. In future work, an additional option could be implemented to specifically allow or prevent the algorithm from building parallel lines.

6. Conclusion

This work gave an introduction to ant colony optimization and how it can be used in the context of medium voltage grids. The presented algorithm was tested and evaluated on a real world cross voltage example and showed good results. It found a solution, which only cost 89% when compared to the cost of the example grid. Furthermore, the crucial triangulation step of the algorithm was examined. The benefits are a reduction of the search space which comes at the cost of not finding valid solutions in certain cases. Finally, a parameter study was conducted to understand more about the influence of certain parameters on the final result and its findings can be used to further improve the performance of the algorithm and its default parameters.

Additional improvements to the algorithm would be to develop a method to fix problematic solutions in a postprocessing step to avoid restarts and therefore speed up the algorithm. Also, the functionality of explicitly allowing or preventing the algorithm from building parallel lines could be implemented. Another way of improving the runtime would be to parallelize the algorithm via the usage of multiple independent colonies. More research is also required in comparing the performance of $APMV$ with other grid planning algorithms (e.g. tabu search, genetic algorithms). This should be done on multiple other real world examples including grids with larger size.

Finally, based on the results of this work, ACO seems to be a promising tool to improve automated planning of electric grids in the future.

Acknowledgements

First and foremost I would like to thank my family who always supported me through the years of my studies. Without them, I would not have been able to come this far. Also I thank Vrishank, Rachit, Jonathan, Erik, Ibrahim and all other Students who worked with me at Fraunhofer ISE and with whom I had a great time in- and outside of work. A big thanks also goes to Janis Kähler for helping me with numerous questions and tasks concerning *PyPSA* and other implementation related issues. Furthermore, I thank Wolfgang Biener, who gave me guidance for my work and who was patient when it took me longer than expected to solve certain tasks. For the last weeks of writing I thank my Uni-Library-Crew Damaris, Mara, Carolin and special guests for brightening my days and motivating me for the final push. And lastly, I would like to thank Prof. Christian Shindelhauer and Prof. Christof Witter for examining this work.

A. Appendix

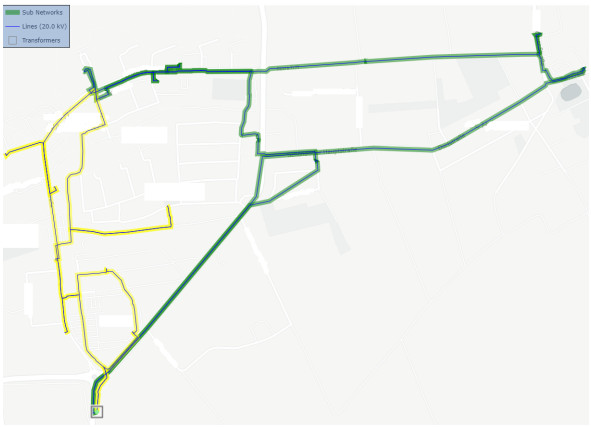

Figure 27.: Solution with a maximum of 8 or 9 buses per ring. Two rings are built (marked yellow and green).

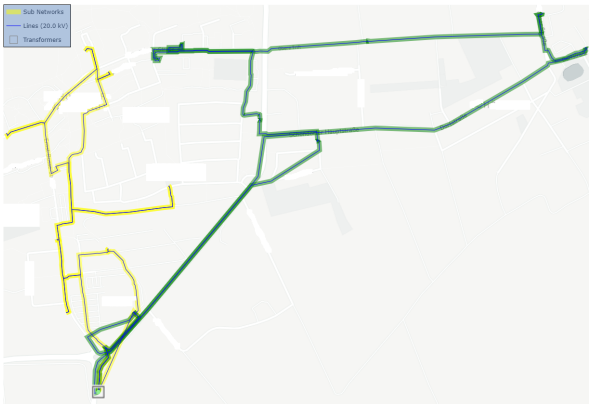

Figure 28.: Solution with maximum of 10 buses per ring.

Bibliography

[1] D. B. Dr. Kemmler, Wünsch, "Entwicklung des Bruttostromverbrauches bis 2030," tech. rep., Bundesministerium für Wirtschaft und Energie, 2021.

[2] T. P. u. E. Bundesnetzagentur für Elektrizität, Gas, "Bericht zum Zustand und Ausbau der Verteilernetze 2021," 2021.

[3] N. Rotering, *Zielnetzplanung von Mittelspannungsnetzen unter Berücksichtigung von dezentralen Einspeisungen und steuerbaren Lasten.* Printproduction Wolff, 2013.

[4] P. Georgilakis and N. Hatziargyriou, "A review of power distribution planning in the modern power systems era: Models, methods and future research," *Electric Power Systems Research*, vol. 121, 04 2015.

[5] R. H. Fletcher and K. Strunz, "Optimal distribution system horizon planning–part i: Formulation," *IEEE Transactions on Power Systems*, vol. 22, no. 2, pp. 791–799, 2007.

[6] G. Celli, S. Mocci, F. Pilo, D. Bertini, R. Cicoria, and S. Corti, "Multi-year optimal planning of active distribution networks," 05 2007.

[7] M. S. Nazar and M. R. Haghifam, "Multiobjective electric distribution system expansion planning using hybrid energy hub concept," *Electric Power Systems Research*, vol. 79, no. 6, pp. 899–911, 2009.

[8] A. M. Cossi, R. Romero, and J. R. S. Mantovani, "Planning and projects of secondary electric power distribution systems," *IEEE Transactions on Power Systems*, vol. 24, no. 3, pp. 1599–1608, 2009.

[9] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[10] E. Bonabeau, M. Dorigo, G. Theraulaz, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. No. 1, Oxford university press, 1999.

[11] M. Dorigo and L. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[12] L. Gebhard, "Expansion planning of low-voltage grids using ant colony optimization," 2021.

[13] J. Zeller, "Planung von Mittelspannungsnetzen mithilfe eines Ameisenalgorithmus," 2021.

[14] J.-L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels, "The self-organizing exploratory pattern of the argentine ant," *Journal of insect behavior*, vol. 3, no. 2, pp. 159–168, 1990.

[15] M. Dorigo and T. Stützle, *Ant colony optimization*. MIT Press, 2004.

[16] W. Schmitting, *Das Traveling-Salesman-Problem: Anwendungen und heuristische Nutzung von Voronoi-Delaunay-Strukturen zur Loesung euklidischer, zweidimensionaler Traveling-Salesman-Probleme*. PhD thesis, Zugl.: Düsseldorf, Univ., Diss., 1999, 2000.

[17] R. John, "Planning of synthetic low voltage networks with geographical constraints," 2021.

[18] T. Brown, J. Hörsch, and D. Schlachtberger, "Pypsa: Python for power system analysis," *arXiv preprint arXiv:1707.09913*, 2017.