Game Theoretic Stable Microgrid Formation in the Electricity Grid

K. Oudshoorn

Technische Universiteit Delft

TU Delft Delft University of Technology

Challenge the future

Game Theoretic Stable Microgrid Formation in the Electricity Grid

by

K. Oudshoorn

in partial fulfillment of the requirements for the degree of

Master of Science
in Computer Science

Supervisor:	Prof. dr. C. Witteveen	
Thesis committee:	Prof. dr. C. Witteveen	TU Delft
	Dr. M. M. de Weerdt,	TU Delft
	Dr. L. M. Ramirez Elizondo,	TU Delft

An electronic version of this thesis is available at
http://repository.tudelft.nl/.

TU Delft Delft University of Technology

ABSTRACT

The current architecture of the power grid is outdated and will not provide the means to deal with the decentralization of energy sources. The smart grid is a newly envisioned architecture for the power grid that should solve the weaknesses in the current grid. One application that is part of the smart grid vision is the microgrid: a coalition of prosumers that is able to operate either in parallel to the power grid or isolated from it.

In this work we investigate how these microgrids should be formed in order to maximize their usability while at the same time preserving the freedom of choice for participants. We apply a game theoretic perspective to simulate the choices of individual users. This results in a problem that combines the notion of core stability and maximization of social welfare. For this problem we provide a complexity proof, followed by a relaxation of core stability, k-stability, by limiting the available knowledge for participants. Finally we produce our own distributed algorithm which can act as a heuristic.

CONTENTS

1

INTRODUCTION

The job of the power grid is to reliably provide electricity to consumers. This also means shielding the consumer from *blackouts* and *brownouts*. Blackouts are events when there is no electricity at all, they are the result of equipment failure, environmental events or even human errors. Brownouts are occurrences of a voltage drop in the supply of electricity (a voltage sag).

Blackouts are, most of the time, caused by a single failure: the tearing of a cable due to stormy weather or perhaps a malfunction at a power plant. Even though there is only a small area where the cause is, the affected area can be quite large. A blackout can even result in a *cascading failure*: an event where one failure leads to a chain of failures.

Sudden loss of electricity can be an annoyance for households: impossibility of watching TV, shutting down of electrical heating. For other areas however, power outages have a more damaging effect. For instance, most public transport is powered by electricity: trains, trams and the subway. Even though blackouts do not occur often, when they do occur, they affect a large number of people. For example, during last years' blackout in Turkey around 70 million people were left without power over a duration of 5-10 hours [The15]. Even though this is an exceptional case, NetBeheerNL [Net12] reported that the average Dutch citizen is deprived of electricity for 20 minutes a year. The situation is worse in the U.S. where, according to Clark [Cla14], customers lose power for an average of 214 minutes per year.

Brownouts are detrimental to the power grid since voltage sags cause devices to work poorly (e.g. dimmed lights). Other devices that are more sensitive to voltage sags (like PC's and microwaves) even need to be reset as a result. For households these events are usually just a nuisance, but, as Bollen [Bol96] states, for industrial processes they lead to a loss of revenue or even dangerous situations.

According to Willis et al. [Wil+16], the current power grid is old-fashioned as its architecture has hardly changed since its creation. When the power grid was

first created, its only usage of electricity was lighting. During this time it was a lot easier to predict the electricity demand. In the present, there are numerous applications for electricity, making it much harder to predict the demand during the day. The grid itself has hardly evolved with this trend. Even though efforts have been made to stimulate certain behaviour of consumers by using pricing mechanisms[1], the grid still has no means to adapt to consumer demand in real time. In order to prevent brownouts from occurring, the grid makes sure to generate more electricity than would be needed in a demand peak. This allows the grid to avoid the need of real time information. In doing so, the grid is also wasting a lot of energy, since these peaks only sparsely occur.

Another disturbing factor for the management of the power grid has been the introduction of distributed energy resources (DER's). Because consumers install their own solar panels and wind turbines, there are occasions where excess energy is fed back into the grid electricity. This transforms consumers into so called *prosumers*: entities that alternate between consuming and producing electricity. As of such, the behaviour of the grid is changing. Energy is no longer produced on a few locations, but rather anywhere in the grid depending on the prosumers' demand, wind speeds, brightness of the sun and other factors. As a result, the current hierarchy structure of the power grid is starting to make less and less sense and the prediction of electricity demand is becoming increasingly harder.

In coming years, it is expected that the demand for electricity will keep on increasing, especially with the upcoming of electric vehicles. Estimations indicate that between 2007 and 2050, electricity consumption will increase by 115% [Tan11]. Given the current infrastructure, the larger energy demands and the increased difficulty of demand prediction the power grid is becoming harder and harder to manage.

Due to this increasing dependency on electricity, it is all the more important to increase reliability and efficiency of the grid. At the same time it is also important to provide means to recover from grid failures as it is infeasible to construct a completely secure grid. The current architecture of the power grid does not have the means to fulfil these needs. Thus it is required to transform the power grid into something that can: *the smart grid*.

1.1. THE SMART GRID

The smart grid is an evolved version of the current power grid. It is not the result of a single change, but instead relies on a number of them. Tanaka [Tan11] provides probably the best description of what a smart grid encompasses:

> "*The smart grid* is an electricity network that uses digital and other advanced technologies to monitor and manage the transport of electricity from all generation sources to meet the varying electricity demands of end-users. Smart grids co-ordinate the needs and capabilities of all generators, grid operators, end-users and electricity mar-

[1] A common used pricing mechanism is making electricity cheaper in the night and weekends.

> ket stakeholders to operate all parts of the system as efficiently as possible, minimizing costs and environmental impacts while maximizing system reliability, resilience and stability."

Transitions towards the smart grid has already begun in recent years with the introduction of smart meters. These allow a two-way flow of information: the supplier obtains information on the demand of the consumers and consumers obtain information about the prices set by the suppliers. Energy suppliers can use this information to properly adjust their power generation or provide incentives to the consumers to divert from their current behaviour. At the same time the consumer can estimate bills and manage their energy consumption accordingly.

Smart meters are, however, only the beginning. [Fan+12] predicts that, ultimately, the smart grid will have the capabilities of autonomous maintenance, recovery of power outages (self-healing), enhanced efficiency and other benefits. An overview of the difference between the existing grid and the smart grid can be seen in Table 1.1.

Existing grid	**Smart grid**
Electromechanical	Digital
One-Way communication	Two-Way communication
Centralized generation	Distributed generation
Hierarchical	Network
Few sensors	Sensors throughout
Blind	Self-Monitoring
Manual restoration	Self-Healing
Failures and blackouts	Adaptive and islanding
Manual check/test	Remote check/test
Limited control	pervasive control
Few customer choices	Many customer choices

Table 1.1: A comparison of the smart grid and the existing grid by Farhangi [Far10]

The smart grid is envisioned to improve the power grid in a lot of areas, but it is not a single invention that allows these improvements. As stated in Tanaka's quote, the smart grid will be the result of combining several technologies. We can therefore assume that there will not be a golden hammer that will allow each of these improvements at once. In this thesis we will focus on how the smart grid will achieve a network structure and counter failures and blackouts.

One idea to minimize damage as a result of failures is to contain the areas where they occur. However, the power grid does not have the capability to do this. After all, this would require locating the cause of the blackout and disconnecting the appropriate section of the power grid. In addition, one cannot blindly remove a section of the power grid as it cannot not guaranteed that the rest of the grid will have enough electricity to continue normally.

A better option would be to define self-sufficient clusters in the grid, which

can remove outgoing connections when faults occur in the grid. Even though no direct action is taken here to contain the failure, the other areas cannot be affected by it since they isolate themselves from the entire grid. This way the majority of the grid will be unaffected by the power outage while the cause of the outage is being solved.

In [LA02], a system was proposed that can do just that: *microgrids*. The idea of microgrids is to sort prosumers into small manageable groups. A microgrid is envisioned to have the ability to isolate itself from the rest the grid, so called *islanding*. By allowing the microgrid to operate as a separate power grid, the prosumers within will be unaffected by blackouts happening elsewhere in the grid. Furthermore, these groups will be managed internally, allowing the grid to view such a group as a single, large prosumer, effectively reducing the amount of entities in the grid that need to be managed. As a result microgrids will simplify management of the grid and provide reliable energy for the prosumers within.

1.2. MICROGRIDS

In general, a microgrid is considered to be a coalition of energy sources, loads and storage devices that is able to operate either in parallel to the existing utility power grid or isolated from it.

Storage devices (e.g. large batteries or flywheels), can be considered to be like a prosumer: they consume energy when charging and produce energy when discharging. Their downside however, is that they have a limit to the amount of electricity they can deliver/store. As these limits have to be taken into consideration during their usage, they complicate any reasoning on microgrids. For this reason we neglect their existence in this thesis.

We therefore define the microgrid as follows:

Definition 1.1. *A microgrid is a coalition of prosumers that is able to operate either in parallel to the power grid or isolated from it.*

Concerning how large microgrids are, this is not set in stone. There are existing examples of various sizes: in Schwass [Sch08] a single hospital was described and in Østergaard and Nielsen [ØN08] an entire island with over 28 thousand inhabitants. Burr, Zimmer, and Douglass also state the following in *About Microgrids*:

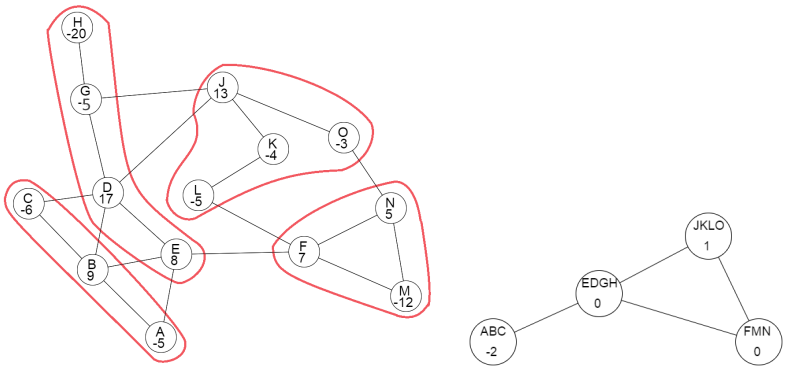
> "Microgrids are defined by their function, not their size".

The exact size of a microgrid is not considered to be important. What is important is what it can achieve.

So, what is the function of microgrids? It was previously indicated that microgrids will simplify management of the grid. A microgrid can be seen from the outside as a single entity that produces or consumes energy. This allows the environment to form trade contracts with the microgrid without having to concern itself with the numerous entities within the microgrid, thus decreasing the amount of management needed (from an outside perspective).

Example. *In figure 1.1 there are two different presentations of a small power grid based on an American Electric Power System [Chr99]. This grid contains different loads and sources, where each entity has the amount of energy it supplies or consumes displayed as a positive or negative number. This grid has been divided into four separate microgrids (in figure 1.1a indicated by the red borders).*

A simplification of the grid is shown in Figure 1.1b. Here, the prosumers inside the same microgrid have been aggregated into one entity. The aggregation maintains the total amount of energy consumed/produced and the external connections. The result is a smaller representation of the same grid.



(a) (b)

Figure 1.1: Two different presentations of a small power grid. On the left is a detailed version with a number of sources and loads with the red borders indicating microgrids. On the right is the aggregated version where only the microgrids and their total energy supply/consumption is displayed.

The other function of microgrids is to provide reliable energy for the prosumers within. On one hand, we can consider this to be achieved because the local management of the microgrid prioritizes the benefit of its members, but the main contribution is a microgrid's ability of islanding.

Islanding is a different name for the act of a microgrid isolating itself from the main power grid. In doing so, prosumers inside the isolated microgrid will only depend on local power production and consumption. They are therefore unaffected by any events that occur on the outside of the microgrid. In the original work on microgrids (Lasseter and Akhil [LA02]), islanding becomes possible by placing separation devices in between the connections to the main power grid. When required, these devices shut down these connections and isolation is achieved.

Blackouts and brownouts, whenever they occur, have been shown to affect a large area even though the cause can be at a single point in the grid. When the cause of such an error is outside of a microgrid, a microgrid can protect its prosumers by isolating itself. The isolation itself does nothing to solve the fault, but since the prosumers in the microgrid are unaffected, damage is mitigated. When the entire power grid is subdivided into microgrids, the islanding of mi-

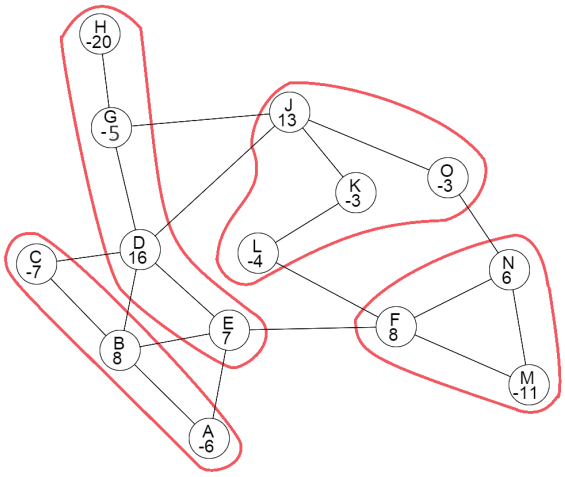


Figure 1.2: A situation of the small power grid where microgrids are unbalanced.

crogrids helps to contain the area of influence of a brownout/blackout to the single microgrid it originates from. Assuming their microgrid is self-sufficient, the prosumers will not notice much of any blackouts/brownouts occurring on the outside.

There is however, a possibility that a microgrid is not entirely self-sufficient. For the energy balance of a microgrid during isolation, three different scenarios can occur:

1. The supply and demand of energy is balanced. This is the optimal case.

2. There is an excess of energy. Producers will have to lower their supply, resulting in a loss of profit.

3. There is a shortage of energy. Consumers will have to lower their demand, limiting their actions.

Apart from the first scenario, there are two cases where someone will be negatively impacted due to isolation. Normally, shortages and excesses can be managed with the rest of the grid by selling or buying energy. When a microgrid is isolated, this option disappears and different tools will need to be used like changing the behaviour of the prosumers within. However, this can have a negative impact on the prosumers, as people are usually not so happy if they are not allowed to watch television.

Defining a balanced microgrid is, however, not enough. The power grid is a dynamic environment. Consumption of electricity is already variable over the day and, with the upcoming distributed energy sources, the production is also starting to become less constant. There is therefore no guarantee that a balanced microgrid will stay balanced over time.

Example. *In figure 1.1, we see the occurrence of the three possible scenario's for a microgrid: there are two microgrids with an overall energy consumption of 0kw, one with a positive amount and one with a negative amount. With the imbalance being relatively close to zero, the microgrids can be considered to be acceptable. However, even minor changes in the grid can disrupt this balance.*

If we consider a group of prosumers to rely on solar energy (A, B, C, D and E) and a different group (F, K, L, M and N) to rely on wind energy. But the weather changes, it becomes more cloudy and the wind speeds up. The solar energy group will notice a decrease in production and the wind energy group an increase. Every prosumer within these groups will have a decrease/increase in power of 1 kW. The result can be seen in figure 1.2. Here there are a number of unbalanced microgrids. $\{A, B, C\}$ *has an accumulated shortage of -5 kW. For* $\{E, D, G, H\}$ *this is -2 kW. On the other hand,* $\{L, K, L, O\}$ *and* $\{F, M, N\}$ *now both have an excess of 3 kW.*

Static microgrids, no matter how well configured, are not a perfect solution. Over time, there will always be someone who is negatively affected by it. A more versatile approach, where energy imbalances could be continuously be avoided, would therefore be better, as that would treat the cause instead of the symptoms. Such a situation can be obtained by occasionally altering the composition of the microgrid. For example, when there is a shortage of energy in a microgrid, prosumers with a large energy consumption could be excluded from the coalition or other energy producers could be included. This means that microgrids would become dynamic entities instead of static.

1.3. DYNAMIC MICROGRIDS

Allowing microgrids to change their composition over time increases their capability to keep their energy production/consumption balanced. This would allow microgrids to exclude or include prosumers according to its current needs.

Changing the composition of a group of consumers in order to obtain a targeted energy consumption is a concept that has been introduced in [Min+10]. Here, consumers are grouped together to allow for an easier demand management. These groups are periodically reorganized to obtain a demand curve within the groups that is nearly constant as can be seen in figure 1.3. As power generation is usually configured to accommodate (possible) peak demand, a more constant consumption results in reduced production costs and creates a more reliable system [Fan+12].

Microgrid reorganization has another benefit besides adjusting the peak demand. It can also be used to maintain self-sufficient microgrids such that when isolation is required, none of the isolated prosumers will have to suffer. As faults can occur out of nowhere and rapidly disrupt the system, there is no time to organize the microgrids properly after the event occurred. By constantly reorganizing the microgrids according to their estimated consumption/production, it will be known beforehand which separation devices should be activated to obtain isolation the moment a fault is observed.

The reorganization of the microgrids does not involve any physical changes to the grid. It merely dictates which separation devices will be activated when

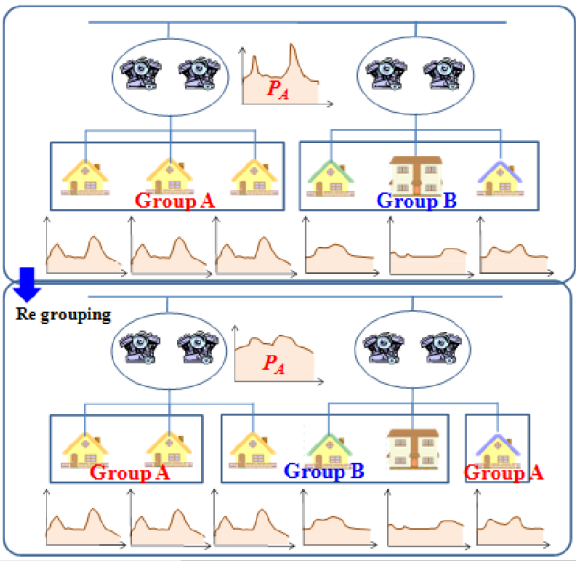


Figure 1.3: Appliance of dynamic microgrids as defined in [Min+10]. Two existing groups of consumers are reorganized to obtain a configuration where the energy consumption is more constant.

isolation is required. As it is easier to balance the supply and demand using additional outside sources/loads, a microgrid will usually operate in parallel with the main grid. No connections will therefore be established or severed during the reorganization itself, this only happens due to the activation of separation devices.

Example. *Continuing from the point where the microgrids were imbalanced (see figure 1.2). We can attempt to rebalance them by altering the microgrid composition. After the reorganization, the composition of figure 1.4 is reached. Here, we have three microgrids with an accumulated energy consumption/production of (nearly) 0 kW.*

Now if a power outage is caused by some error at E, the microgrids will react by removing all outgoing connections (e.g. $E-G$ and $B-C$). This means $\{F,N,M,O\}$ will be on its own for some time though be it with no excess or shortage of electricity. Had it been the organization of Figure 1.2, the microgrid $\{F,N,M\}$ would have had an excess of electricity during this isolation.

In order to be able to reorganize the microgrids, the power grid will need the ability to break connections at multiple places in the grid. As any formed microgrid has to be able to isolate itself, a microgrid that does not have a separation device on all outgoing connections is meaningless. Thus to be able to reorganize microgrids, a larger amount of separation devices is necessary in the grid than it would be the case for static microgrids.

On the other hand, it is not a must to have a separation device in between every connection. Entities that cannot disconnect from each other can be ag-

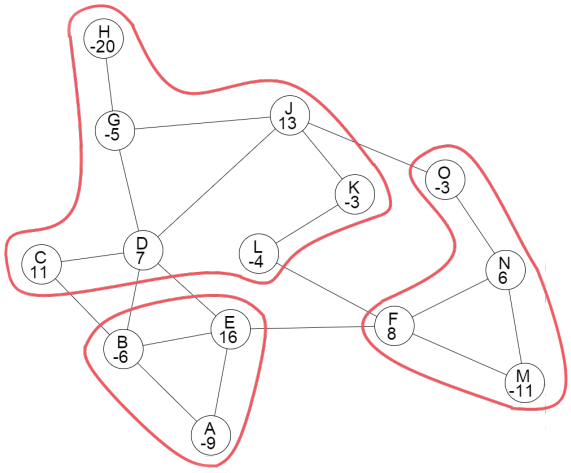


Figure 1.4: A rearranged version of the unbalanced power grid. Here, microgrids are once again (nearly) balanced.

gregated and considered as one entity instead, similar to the aggregation of Figure 1.1b.

With the introduction of dynamic microgrids, a microgrid is no longer a structure that has been planned, designed or revised by a team of experts. Instead, it is a continuously changing structure. If the reorganization of the microgrids is not done properly, the newly formed microgrids could turn out worse than what they used to be or might even become unusable. The formation of dynamic microgrids should therefore be handled with care.

1.4. PROBLEM STATEMENT

Microgrids can be a great asset for the power grid in terms of management and reliability. At the same time, they are delicate structures that cannot be created without some sense of thought. Making microgrids dynamic improves the benefits that they provide, though that does not mean that any reorganization will result in an improvement and could even lead to a deterioration.

In this thesis we will research how we can compute beneficial reorganizations of microgrids. On the basis of the work of Mine et al. we will determine what "beneficial" is. Our research is solely interested in the method of computation, not the hardware and alterations of the power grid that are required to realize these reorganizations.

In this thesis we will investigate how it can be ensured that the reorganization of microgrids will be beneficial. We hope to provide some insight on this subject by solving our main problem:

What are suitable techniques to form dynamic microgrids?

The suitability of a technique is determined by a number of requirements:

- *Scalability*: as a power grid can easily consist out millions of entities, microgrid formation can be quite difficult, but this should not result in the formation to take years to calculate.
- *Feasibility and optimality*: the resulting microgrid distribution should be feasible in practice and optimized (what feasible and optimal is will be discussed in Chapter 2).
- *Preservation of freedom of choice*: eventually microgrids are being used and made by individuals, the fact that microgrid assignment could occur through an algorithm does not remove the ability of those individual to agree or disagree with the assignment.

1.5. OUTLINE

Admittedly the requirements might seem a bit general, but during the remainder of this thesis they will be clarified. The next chapter will start with an investigation of the closely related work of Mine et al. [Min+10]. This will help us define the constraints and preferences of microgrids.

With our goal being formulated, the focus will shift towards related work on comparable problems, outside of the power grid research area. After examining this work, a consideration will be made on what is still required in order to solve our initial problem. This will lead to the formulation of a number of research questions.

Chapter 3 will attempt to answer these questions by providing new work in the form of definitions algorithms and proofs. Some questions might only be hypothetically answered in case no theoretical proof can be provided. An empirical investigation will be conducted in Chapter 4 in order to test these hypothesises.

Finally, in Chapter 5, all obtained results will be summarized and discussed in contrast to the original problem statement in order to conclude this thesis.

2

BACKGROUND

In the previous chapter, the problem of forming microgrids has been introduced. While we have a general idea of what this problem involves (partitioning a network of prosumers into coalitions that optimizes the energy balance), we lack a formal definition. The work closest related to the formation of dynamic microgrids we have found so far is that of Mine et al. [Min+10]. As of such this work will be used as a starting point of our investigation. We will explore the interesting points of this work together with its downsides and use that to define the constraints and goals of microgrid formation. Next, the closely related problem where players try to form coalitions for increased benefit, Coalition Structure Generation and its related work will be discussed in Section 2.4 followed by a more specific version that is constrained by the topology of networks in Section 2.5. Afterwards, a different problem is discussed where the overall quality of the coalitions no longer considered. Instead the problem is centred around the rationality of players, meaning players care less about others and more about the benefit they obtain. This chapter will then finally be wrapped up by a discussion of the found work and the formulations of the research questions that remain.

2.1. GROUPING OF ELECTRICITY CONSUMERS

As stated before in Section 1.3, Mine et al. propose to repeatedly alter the composition of consumer groups in order to reach a targeted energy consumption that minimizes fluctuations in demand.

First of all they calculate for each group its target consumption[1]. Each consumer has a predicted amount of energy consumption over a certain timespan. This knowledge is used to divide the consumers over the groups in such a way that the total absolute difference between the target consumption and the pre-

[1] According to the authors, the target consumption of a group is calculated using the history of power consumption of that group, though exact details have been omitted.

dicted consumption is minimized. In [Min+10], the market of Japan is considered where electricity is sold in intervals of 30 minutes. As of such, the considered timespan is of 30 steps of a minute. More formally, this problem can be described as an *Optimized Grouping* problem.

Optimized Grouping: *Given a set of bins M, with for every $j \in M$ a target sum at time step t $G'_j(t)$, a set of individuals V, where each $i \in V$ has an integer associated at time step t, $H_i(t)$. Define $G_j(t) = \sum_{i \in j} H_i(t)$. Divide V over M such that $\sum_{j=1}^{M} \sum_{t=0}^{30} (G_j(t) - G'_j(t))^2$ is minimized.*

The Optimized Grouping problem is solved using a mixed integer linear program (MILP). In this program, the distribution of V over M is indicated by a binary matrix X.

$$X = \begin{pmatrix} X_{1,1} & \cdots & X_{1,m} \\ \vdots & \ddots & \vdots \\ X_{n,1} & \cdots & X_{n,m} \end{pmatrix} \tag{2.1}$$

Where $m = |M|$ and $n = |V|$. In X, every column represents a bin and every row an individual. If an $X_{ij} = 1$ then that means $i \in V$ is part of $j \in M$. Furthermore, it is enforced that an individual must be part of exactly one bin. This means that exactly one element in every row should be set to 1 (since X is a binary matrix, only 0 or 1 is an appropriate value). This boils down to the following MILP:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{M} \sum_{t=0}^{30} (G_i(t) - G'_i(t))^2 \\ \text{Subject to} \quad & \sum_{m=1}^{M} X_{nm} = 1 && \forall n \in V \\ & X_{ij} \in \{0,1\} && \forall i, j \end{aligned}$$

As explained, the optimization function is used to obtain a group structure where each j has an expected sum $G_j(t)$ that approaches its target sum $G'_j(t)$ as close as possible over a timespan of 30 steps (hence the summation of $t = 0$ to 30). The constraint function deals with the amount of different bins an individual can be in.

Example. *Consider the consumers of Figure 2.1 $\{A, C, G, H, K, L, M, O\}$, with their respective consumption amount $H = \{5, 6, 5, 20, 4, 5, 12, 3\}$ (in this example we will negate any changes over time). Assume, the energy consumption does not change overtime. We would like to form four groups where energy production and consumption is balanced. Thus we formulate $G' = \{9, 25, 13, 12\}$. Then if we solve our*

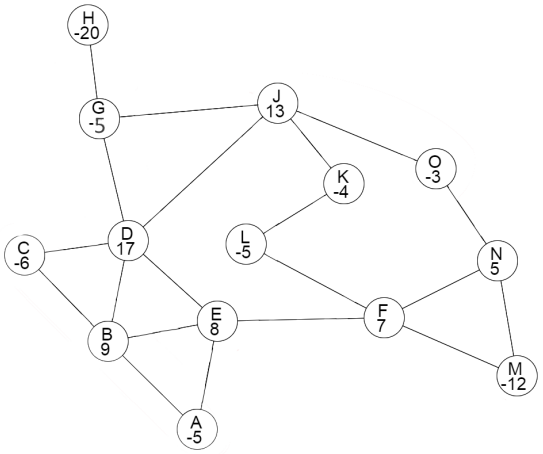


Figure 2.1: A power grid representation before any micorgrids have been formed. Prosumers are connected by electricity cables (the connections between nodes) and have a energy production/consumption indicated by the number inside the nodes.

MILP, we find:

$$X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.2}$$

With $G = \{9, 25, 14, 12\}$, *hence* $\sum_{j=1}^{M} (G_j - G'_j)^2 = 1$. *Translating* X *to a more readable set of coalitions we get* $\{\{A, K\}, \{G, H\}, \{C, L, O\}, \{M\}\}$. *Including the producers that were attached to the coalitions we get* $\{\{A, B, K\}, \{D, E, G, H\}, \{C, J, L, O\}, \{F, M, N\}\}$

The Optimized Grouping problem seems very similar to the problem of forming dynamic microgrids. There are however, some important differences that show why we cannot apply this approach directly:

- Negligence of topology
- Negligence of producers
- Arbitrary target consumption
- Fixed amount of microgrids

Each of these differences will be discussed to give a clearer view of their meaning.

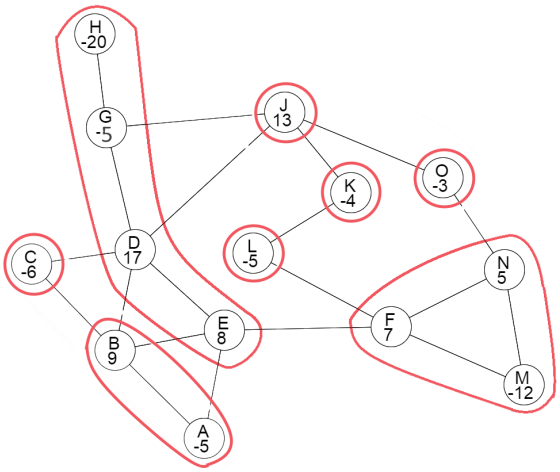


Figure 2.2: A small power grid divided up into microgrids according to the partitioning $\{\{A,B,K\}, \{D,E,G,H\}, \{C,J,L,O\}, \{F,M,N\}\}$. Due to not every partition being connected, the effective partitioning is in fact $\{\{A,B\}, \{K\}, \{D,E,G,H\}, \{C\}, \{J\}, \{L\}, \{O\}, \{F,M,N\}\}$

2.1.1. NEGLIGENCE OF TOPOLOGY

First of all, the topology of the grid seems to be disregarded. It is implicitly assumed that the grid spans a complete graph. In case a microgrid isolates itself from the rest of the grid, it severs all outside connections. This means that in order to be able to share electricity within the microgrid it needs to span a connected graph by itself. In the MILP, this property is not guaranteed unless the whole network is a complete graph. Reality is, however, far from this situation. In a survey on the properties of the grid infrastructure by Pagani and Aiello [PA12] it was shown that an entity is, on average, connected to roughly three other entities. It is therefore unlikely that a random partition from the grid will span a connected graph.

Example. *In the previous example, one of the formed groups was* $\{C,L,O\}$. *If this were a microgrid that isolates itself by closing off all external connections*[2] *we would end up with three separate prosumers instead of one island. If this idea is extended to the whole partitioning, a collection of microgrids is found that is rather unbalanced in contrast to the proposed solution (see Figure 2.2).*

2.1.2. NEGLIGENCE OF PRODUCERS

The second difference is the negligence of power suppliers. Energy producers are also part of the grid and since they are becoming a less centralized entity, they are important for ensuring the connectedness of microgrids. Furthermore, consumers are not the only ones with fluctuations. Not every power source can provide a constant amount of energy (think of windmills and solar panels).

[2] An external connection is a connection between two prosumers of different microgrids

Therefore, it is important to know which suppliers are to be within the microgrid, instead of assuming a giant energy pool.

2.1.3. ARBITRARY TARGET CONSUMPTION

This brings us to the third difference: the arbitrary target consumption of each group. When producers are also able to switch groups, target consumption is a meaningless value. It would suffice to say that the consumption of a microgrid should be close to equal to its production. On a side note, it is also unclear how targeted consumption is calculated by Mine et al. as they only shortly mention that it is based on the history of the existing groups.

2.1.4. FIXED AMOUNT OF MICROGRIDS

The last difference with Optimized Grouping is that, there, the number of bins is fixed. With the removal of target consumption, there is no reason to keep the number of microgrids fixed. This allows for more freedom and solutions with a possibly more balanced energy production/consumption.

Clearly, the formation of dynamic microgrids is not as straightforward as it seems. It is, therefore, important to define what the dynamic microgrids are trying to achieve and what the constraints of a valid solution are. For this reason we need to dive a bit deeper into the definition of a microgrid and investigate what a valid microgrid is and when it is beneficial for its prosumers.

2.2. DEFINING THE MICROGRID

In the previous section we have noted several aspects that could collide with the very idea of what a microgrid is (network topology being one of them). For clarity it is needed to formulate constraints for microgrids, then we will have a good view of what a legitimate microgrid is.

A legitimate microgrid is however, not per se a beneficial one. For one, it is not a requirement for a microgrid to have a balanced energy production and consumption, though it is highly preferred. There are certain preferences for the composition of a microgrid and in order to find the optimal one, these preferences need to be defined.

2.2.1. CONSTRAINTS OF MICROGRID FORMATION

The formation of microgrids is constrained by a number of factors. One cannot simply group together a bunch of prosumers and call them a microgrid. There are rules for this process and it is important to know what they are.

First of all, the prosumers within a microgrid have to be connected with each other, either directly or through a path of other prosumers of the same microgrid. In practice, a prosumer is not directly connected to everyone else in the grid and when microgrids are in island-mode, energy can only be transported over connections within the microgrid. Thus, When there are sources and loads that are not interconnected within this island (as in figure 2.2), energy cannot be shared between these entities and the whole microgrid idea becomes obsolete.

Furthermore, prosumers will have the ability to choose whom to affiliate with. Since, it hard to predict this kind of behaviour, we will assume that prosumers are rationally selfish. This means that they will want what benefits them the most[3] and do not decide on factors like a gut feeling. It is important to think about how these prosumers would act in consideration to the rest of the grid. Selfish prosumers will not accept a given configuration if they know that there is a better one available for them.

With the knowledge of these constraints the eventual formed microgrids should be valid. However, constructing viable microgrids is not the only thing we are concerned about. It is also a goal to form convenient microgrids. A valid microgrid does not guarantee that it is advantageous. There are microgrids that are more beneficial than others. Some can provide more secure energy, others have a larger amount of renewable resources. There are a number of factors that determine the quality of a microgrid.

2.2.2. PREFERRED PROPERTIES OF MICROGRIDS

Microgrids have the purpose of benefiting prosumers, but some can do this better than others. Before it can be determined when a microgrid is better than another, the preferred properties should be indicated. Combining the evaluation of these properties we can form a function that can give an indication of how good a given microgrid is. The idea of this function is that it could be substituted in order to adjust the evaluation of microgrids (without having to change the algorithm).

In the work of [Min+10], consumers were organized based on their estimated demand trend $G_j(t)$ for a given time space. As was already stated in the Section 2.1.2, this approach neglects the existence producers. In order to make a self-sufficient microgrid, prosumers should be gathered such that the estimated demand and supply trend closely matches each other.

In this thesis we simplify the representation of the demand/supply trend over time to a single number. Admittedly, this is an inaccurate representation of the consumption/production of a prosumer as it does not provide any information about a prosumer's behaviour over time. However, switching between the two representations will only influence the difficulty of modelling the prosumers, not the way microgrids will be organized.

Even though the grid consists out of prosumers, it does not take away that there are producers and consumers in the grid. Both have a different preference when it comes to available energy in the microgrid. Consumers generally want a surplus of energy so they have no limits in their behaviour. Simply speaking, they want to be able to turn on the tv whenever they want. Producers, on the other hand, want to sell as much energy as they can produce. It is therefore more in their favour when the demand is higher than the supply.

From our point of view it suffices to say that microgrids should have a balanced demand and supply, since that is in everyone's interest. More formally, it means that for a microgrid $X \subseteq V$, where each prosumer $i \in V$ has an energy

[3] A description of what is beneficial will be given at section 2.2.2

consumption/production r_i, we can calculate the total supply $S(X) = \sum_{i \in X,\, r_i > 0} r_i$ and total demand $D(X) = -\sum_{i \in X,\, r_i < 0} r_i$. The balance value can then be calculated by:

$$u_b(X) = \frac{\min\{S(X), D(X)\}}{\max\{S(X), D(X)\}} \tag{2.3}$$

The closer $u_b(X)$ is to 1, the better the energy balance in X is. The minimum of the demand and supply is divided by the maximum in order to stay indifferent to whether demand or supply is larger. Ratio is believed to be more suitable than difference for the formula of $balance$ as this allows the obtained utility to be proportionate to the relative power balance. A difference of 5 MW has a larger negative impact in a microgrid with a total demand of 10 MW than in one with 100 kW.

Another preferred property of microgrids is safety of errors. Isolation keeps the prosumers in a microgrid safe from occurring on the outside, but it will not help when errors are occurring inside the microgrid. The safety of a microgrid is based on factors such as the type of power sources or the age of the electricity lines present. As these kind of factors are quite hard to model, we have chosen for a different one to represent the safety of a microgrid: its size.

Microgrids should be small in comparison of the power grid. If we were to assume that every location in the grid is equally prone to faults, the probability of a single prosumer being the origin of a fault during such an event, is $\frac{1}{n}$, with n being the total number of prosumers in the grid. Therefore, for a microgrid X, the probability of the fault originating from one of the prosumers of X is given by $\frac{|X|}{n}$. The negation of that, the probability of a fault occurring outside of X is given by $1 - \frac{|X|}{n}$.

When a fault occurs outside of the microgrid, it can protect itself by islanding. To obtain safety of errors, the probability of a fault occurring outside of the microgrid should be preferably large. Thus the safety of a microgrid X is somewhat represented by its smallness:

$$u_s(X) = 1 - \frac{|X|}{n} \tag{2.4}$$

By combining $u_b(X)$ and $u_s(X)$, a function can be created that can evaluate the relative value of a microgrid. As these properties are not necessarily equal in importance with respect to each other, they are both weighted by a variable w:

$$u(X) = w \cdot u_s(X) + (1 - w) \cdot u_b(X) \tag{2.5}$$

$u(X)$ always returns a value between 0 and 1, which allows for an easy judgement of good and bad coalitions (e.g. $u(X) = 0.98$ would be very good and $u(X) = 0.2$ would be poor).

The only problem is that both u_s and u_b provide a linear evaluation of the coalitions. In our opinion, a coalition A that is already balanced ($u_b(A) = 0.85$) has less to gain from improving that balance than a less balanced coalition B ($u_b(B) = 0.5$). A can already be considered pretty self-sufficient but B is not, thus

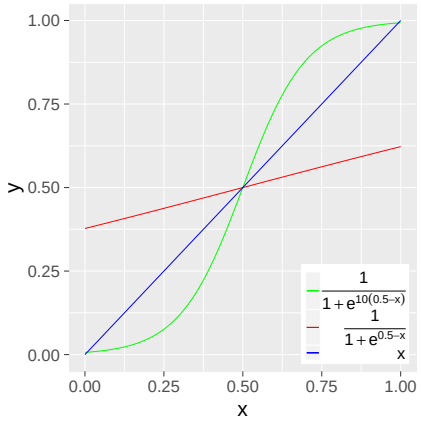


Figure 2.3: Plot of the sigmoid function $\sigma(x) = \frac{1}{1+e^{10\cdot(0.5-x)}}$. It is a slightly modified version of the standard sigmoid function, where the turnover point $y = 0.5$ has been shifted to $x = 0.5$ and the curve occurs in a shorter range. The obtained function is better suited for evaluating values $0 \leq x \leq 1$.

an improvement in balance of 0.1 for both A en B improves the ability to be self-sufficient more for B than it does for A, thus it should provide more extra utility for B than for A. On the other end for the spectrum (close to 0), very imbalanced coalitions are unlikely to be self-sufficient thus an improvement in balance of 0.1 should have a small effect on the utility.

The same reasoning can be applied to the smallness of coalitions as very small coalitions can be considered desirable and coalitions close to the size of the network way too big. The closer the property values are to their limits, the less the differences should matter. To establish this behaviour, a sigmoid function is used (well known for its S-shaped curve).

The standard sigmoid function $\frac{1}{1+e^x}$ has its turning point of 0.5 at $x = 0$. Our goal is to use this function for $x \in (0,1)$ therefore this turning point has to shift to $x = 0.5$ and a small adjustment is necessary: $\frac{1}{1+e^{0.5-x}}$. Furthermore, in order to reach the asymptotes earlier a multiplier of 10 is used. This results in the final σ-function:

$$\sigma(x) = \frac{1}{1+e^{10\cdot(0.5-x)}} \tag{2.6}$$

The behaviour of this function and its alternatives can be seen in Figure 2.3.

With the addition of the sigmoid function, the frame for the utility function is set. However, the value of the weight for balance and smallness has not yet been determined. One thing that can be said is that balance is more important than smallness. The main reason for the coalitions is to obtain a proper balance, smallness is mostly there to make sure that coalitions do not cover the whole network just to obtain a slightly better balance.

Smallness deters the occurrence of extreme size increases for small increase in utility. But it should not be used to justify a small imbalanced coalition. In our opinion, the minimum balance that would overrule the usage of an entire net-

work with perfect balance is around 0.6 . This would probably provide enough means to become temporarily self-sufficient in case of emergencies, though be it far from ideal. Combined with the perfect smallness the utility of such a coalition should be about the same as that of a coalition with perfect balance and and minimal smallness. This leads to the following equation:

$$w \cdot \sigma(1) + (w-1) \cdot \sigma(0.6) = w \cdot \sigma(0) + (w-1) \cdot \sigma(1)$$
$$w \cdot 1 + (w-1) \cdot 0.75 \approx w \cdot 0 + (w-1) \cdot 1$$
$$w \approx 0.2$$

We estimate a proper value for the weight to be $w = 0.2$, resulting in the final utility function:

$$u(X) = 0.2 \cdot \sigma(u_s(X)) + 0.8 \cdot \sigma(u_b(X)) \tag{2.7}$$

With this function we have obtained a means to evaluate how well a single microgrid is. However, how does one assess the quality of multiple microgrids? Is a small amount of elite microgrids better than a large amount of mediocre ones? These questions need answers before we can investigate how the formation should be performed.

According to Gittoi [Git] coalition theory deals with the analysis of one or more groups of agents, called coalitions. This involves evaluating individual coalitions, but also a group of coalitions. The ideas of coalition theory can therefore also be applied to the formation of microgrids, which are in essence a groups (microgrids) of agents (prosumers) as well.

2.3. COALITION THEORY

Coalitions are generally formed when people have something to gain from it. In some cases they need the help of others to achieve a certain goal and in other cases they can achieve their goal more efficiently with the help of others.

Example. *Using the state of prosumers in the power grid of Figure 2.1, we can consider the situation of prosumer M. M is currently consuming 12 kW. When there occurs a blackout and M acts like a microgrid, isolating himself from the grid, M will be a single consumer with no power to consume. Therefore, in order to stay safe during blackouts he will need the help of producers. At the same time, producers like F have an incentive to help out consumers, or they will not be able to sell their energy during isolation. Thus, both producers and consumers have a common interest to form coalitions with each other. However, one can wonder: who should form a coalition with whom in order to maximize the obtained benefits? Will any kind of coalition work? What will happen if a new prosumer joins the grid? These are the kind of questions coalition theory concerns itself with.*

2.3.1. UTILITY OF COALITIONS

A coalition can provide benefits for a person. As one coalition will be more beneficial than another, a preference for certain coalitions can be formed. Such a preference for coalitions can be expressed in a linearly ordered way. If P_M is the

collection of preferences for all coalitions that include M then part of it could look like this: $P_M(\{M,F,N\}) > P_M(\{M,F\}) > P_M(\{M,N\}) > P_M(M)$. As such, it can be stated that M's preference for the coalition $\{M,F,N\}$ is larger than that for $\{M,F\}$.

However, when we start to include coalition possibilities with more and more players, the amount of possible coalitions becomes larger and larger. There are occasions where the amount becomes so large that it is infeasible to maintain a collection of preferences for all coalitions from the viewpoint of all players. For these cases an evaluation function is more appropriate. Thus instead of having a set P_M, there would be a function u_M that can link a value to a coalition that is provided as input. $u_M(\{M,F,N\})$ could for instance depend on the net energy and the size of the coalition (an indication of how stable energy supply will be). the higher this value is, the more robust the energy supply is expected to be and the more preferred that coalition is for M.

Definition 2.1. Utility *represents the motivations of players. A* utility function *assigns a number for every possible coalition with the property that a higher number implies that the outcome is more preferred. [Sho05]*

Knowledge on the utility of coalitions is useful in different ways. As explained, one of these is the ability to determine a preference order for coalitions. Another benefit is the ability to evaluate a set of multiple coalitions, not from the individual's point of view, but from the perspective of the greater good, known as *social welfare*.

2.3.2. SOCIAL WELFARE

Coalitions are not always used for the purpose of improving the utility of selfish individuals, they can also be used to improve the utility of a whole environment. The main reason to use microgrids in the power grid is to provide reliable energy to prosumers. This means that, for the evaluation of the system, it is more important to measure the utility of the entire system instead of deciding whether a single prosumer was able to maximize its utility or not.

As described in [TW04], there are multiple ways to determine social welfare: using the sum of utility for each player, the sum of player utilities and variances and the utility of the worst of player. In this thesis we use the following definition:

Definition 2.2. *Given a set of coalitions* $P = \{P_1, \ldots, P_k\}$ *and a utility function* $u(X)$*, the* social welfare *of* P *is given by* $\sum_{P_j \in P}(|P_j| * u(P_j))$.

Maximizing social welfare is an important goal in order to improve the situation for everyone involved. However, coalitions are non-overlapping, meaning that if someone joins a coalition, he will have to leave his current one. This will have an effect on both coalitions. Coalition formation is not a straightforward task as we will see in the next section.

2.4. COALITION FORMATION TECHNIQUES

Definition 2.3. *Given a set of players $N = \{1,2,...,n\}$ a coalition structure is a set of disjoint subsets $P = \{P_1, P_2, ..., P_m\}$ that partitions V: $\bigcup_{j=1}^{m} P_j = V$, $P_i \cap P_j = \emptyset$ with $i \neq j$.*

Coalition Structure Generation (CSG) focuses on creating a structure such that social welfare is maximized. It has been shown to be a very time consuming problem, since the amount of different coalition structures is so large. An example can be seen in Figure 2.4. Here we can see a Coalition Structure Graph, showing all the different structures that are possible. The connecting lines between the combinations show how we can transfer from one coalition structure to the other.

Coalition Structure Generation (CSG): *Given a set of players V and a utility function $u(X)$. Find a partitioning $P = \{P_1, \ldots, P_m\}$ of V such that $\sum_{P_j \in P}(|P_j| * u(P_j))$ is maximized.*

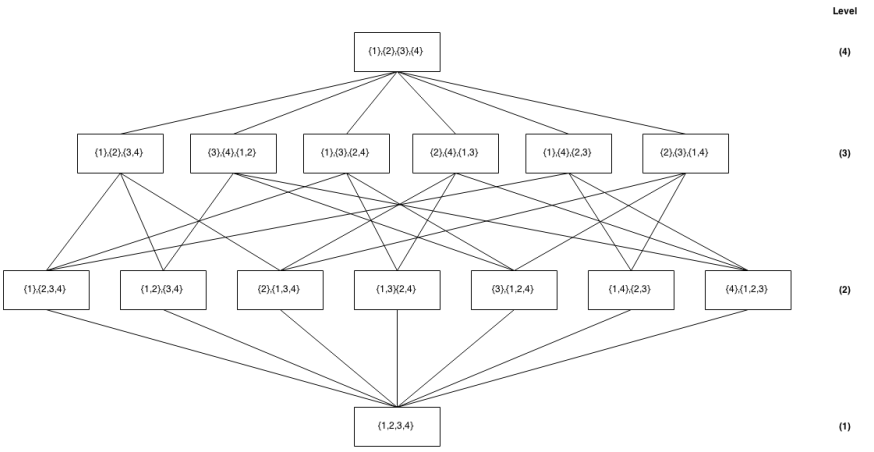


Figure 2.4: Coalition Structure Graph for a game with 4 players[San+99]. Represents all possible coalition structures that can be formed with 4 players. Lines indicate a single merge/split move to transition from one structure to the other.

Consider the number of different ways a set of size n can be partitioned. We can partition the set into one subset, or into two disjoint subsets all the way to n subsets. In the first case the single subset will contain all the elements and in the latter, every subset will contain a single element. Both of these options can only be formed in one way. But with any intermediate amount of subsets the number of possibilities is less straightforward.

Definition 2.4. *The number of ways to partition a set of n objects into i, non-empty sets is given by $Z(n,i)$, the Stirling number of the second kind (or Stirling partition number)[GKP89].*

$$Z(n,i) = iZ(n-1,i) + Z(n-1,i-1) \tag{2.8}$$
$$Z(n,n) = 1 \tag{2.9}$$
$$Z(n,1) = 1 \tag{2.10}$$

As explained before, the base cases consist of Z(n, n), the case where all subsets contain one element, and Z(n, 1), the case where one subset contains all elements. To comprehend the main recurrence formula, one needs to understand how this is obtained from the results of the smaller set. By increasing the set size by one, the new element either needs to be added to existing subsets, or added as a single subset. By adding the element to existing subsets, the amount of subsets does not increase. Since there are i different subsets, $Z(n-1,i)$ can be changed n times. Thus we reach the first term $iZ(n-1,i)$. The second term considers the addition of a subset with solely the new element and therefore counts all the known possibilities for $i-1$ subsets. Hence $Z(n-1,i-1)$ is used.

Using this, we can find the total amount of possible partitions. This is called the Bell Number $B(n)$ (see Figure 2.5). With

$$B(n) = \sum_{i=1}^{n} Z(n,i) \tag{2.11}$$

In [San+99] it has been shown that $B(n) \in \omega(n^{\frac{n}{2}})$ and $B(n) \in O(n^n)$. This means that the growth of the Bell Number sequence is super exponential[4]. Therefore, it can be concluded that the amount of possible coalition structures is exceptionally large.

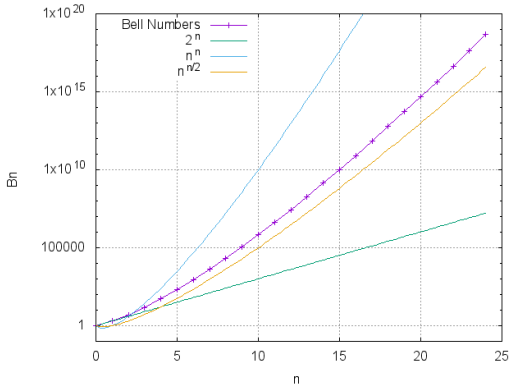


Figure 2.5: The sequence of Bell Numbers [Slo16] compared to 2^n, n^n and $n^{n/2}$.

The most notable achievements in the literature are that of [San+99] and [RJ08]. [RJ08] constructed a dynamic programming algorithm that can find the

[4] A super exponential function grows faster than an exponential function

optimal coalition structure in $O(3^n)$ and [San+99] proved that for a bounded approximation algorithm, at least 2^{n-1} comparisons are needed. While such an approximation is certainly better than checking all combinations, it is still somewhat unusable for something as large as the power grid, where the number of individuals can easily be larger than a thousand ($2^{999} > 10^{80} \approx$ number of fundamental particles in the observable universe).

2.4.1. EXISTING METHODS

In order to form the coalitions, there is the possibility of just letting the agents sort it out on their own. In most cases this will result in the usage of a Best-First decision. Each agent will propose the coalition that is most suitable for him. But when agents have different needs and agendas, chances are you will end up in a situation where agents will keep on changing their coalition because there is always a group of agents that can find a better coalition.

As a rule of thumb, coalition formation usually happens in 3 steps [San+99]:

1. *Coalition Value Calculation*: For each possible coalition, the value of that coalition is calculated. This value can then be used as an indication of how well that coalition will perform, should it be formed. These coalition values can then be used to decide which should form the coalition structure.

2. *Coalition Structure Generation*: Using the coalition values, the coalition structure is decided. Usually the goal is to maximize the social welfare of the system.

3. *Pay-off Distribution*: After coalitions have been formed, there is still the question whether they are stable. A coalition structure with maximized social welfare is no guarantee for stability. However, as shown in [Vin+12], stability can be achieved by distributing the pay-offs properly.

Due to the large amount of possibilities (see Section 2.4), the most computationally intense of these steps is Coalition Structure Generation (CSG). That is probably also the reason the focus of the literature is on that part of coalition formation.

There have been a number of approaches to coalition structure generation, both exact and approximative. Some are more useful in relation to the view of dynamic microgrids, others are not. Here we will discuss a number of them and whether they can be applied to forming microgrids or not.

2.4.2. EXACT ALGORITHM

Rothkopf, Pekec, and Harstad [RPH98] proposed an exact algorithm for CSG using dynamic programming. The worst-case runtime of this algorithm is $O(3^n)$. Later, an improvement on this algorithm was proposed in [RJ08], lowering the amount of needed operations as well as the amount of memory needed. The resulting decrease was at most a factor 3, thus not being able to significantly lower the bound on the runtime.

size	Coalition	The evaluations that are performed before setting f_1 and f_2		f_1	f_2
1	{1}	u({1}) = 30		{1}	30
	{2}	u({2}) = 40		{2}	40
	{3}	u({3}) = 25		{3}	25
	{4}	u({4}) = 45		{4}	45
2	{1,2}	u({1, 2}) = 50	$f_2[\{1\}] + f_2[\{2\}] = 70$	{1} {2}	70
	{1,3}	u({1, 3}) = 60	$f_2[\{1\}] + f_2[\{3\}] = 55$	{1, 3}	60
	{1,4}	u({1, 4}) = 80	$f_2[\{1\}] + f_2[\{4\}] = 75$	{1, 4}	80
	{2,3}	u({2, 3}) = 55	$f_2[\{2\}] + f_2[\{3\}] = 65$	{2} {3}	65
	{2,4}	u({2, 4}) = 70	$f_2[\{2\}] + f_2[\{4\}] = 85$	{2} {4}	85
	{3,4}	u({3, 4}) = 80	$f_2[\{3\}) + f_2[\{4\}] = 70$	{3, 4}	80
3	{1,2,3}	u({1, 2, 3}) = 90	$f_2[\{1\}] + f_2[\{2,3\}] = 95$	{2} {1, 3}	100
		$f_2[\{2\}] + f_2[\{1,3\}] = 100$	$f_2[\{3\}] + f_2[\{1,2\}] = 95$		
	{1,2,4}	u({1, 2, 4}) = 120	$f_2[\{1\}] + f_2[\{2,4\}] = 115$	{1, 2, 4}	120
		$f_2[\{2\}] + f_2[\{1,4\}] = 110$	$f_2[\{4\}] + f_2[\{2,3\}] = 115$		
	{1,3,4}	u({1, 3, 4}) = 100	$f_2[\{1\}] + f_2[\{3,4\}] = 110$	{1} {3, 4}	110
		$f_2[\{3\}] + f_2[\{1,4\}] = 105$	$f_2[\{4\}] + f_2[\{1,3\}] = 105$		
	{2,3,4}	u({2,3,4}) = 115	$f_2[\{2\}) + f_2[\{3,4\}] = 120$	{2} {3, 4}	120
		$f_2[\{3\}] + f_2[\{2,4\}] = 110$	$f_2[\{4\}] + f_2[\{2,3\}] = 110$		
4	{1,2,3,4}	u({1, 2, 3, 4}) = 140	$f_2[\{1\}] + f_2[\{2,3,4\}] = 150$	{1,2} {3,4}	150
		$f_2[\{2\}] + f_2[\{1,3,4\}] = 150$	$f_2[\{3\}] + f_2[\{1,2,4\}] = 145$		
		$f_2[\{4\}] + f_2[\{1,2,3\}] = 145$	$f_2[\{1,2\}] + f_2[\{3,4\}] = 150$		
		$f_2[\{1,3\}] + f_2[\{2,4\}] = 145$	$f_2[\{1,4\}] + f_2[\{2,3\}] = 145$		

Table 2.1: Example from [RJ08] on how the DP-algorithm computes the tables f_1 and f_2.

The idea of the algorithm is to construct a path through the coalition structure graph, starting from the grand coalition, towards the optimal node. Movement through the coalition structure graph can be seen as the splitting of a coalition (merging of two coalitions would be a backwards movement).

Construction of the path is achieved by building two separate tables: f_1 and f_2. In f_1 the most beneficial split is saved and in f_2 the maximum value achievable with that split. The most beneficial split is determined using already filled in values in f_2 for subsets of the current coalition and the value for the coalition itself.

Take for example the coalition {1, 2} in Table 2.1. The value of this coalition is 50 whereas the combined value of the separate coalitions {1}, {2} is larger than that: 70. Thus whenever a node with the coalition {1, 2} is reached in the coalition structure graph, the decision will be made to split into {1},{2}.

When both tables have been filled, the path can be constructed. Starting with the grand coalition {1, 2, 3, 4}, table f_1 returns {1, 2}, {3, 4}. Following these coalitions, the optimal coalition structure is obtained: {{1},{2}, {3, 4}}.

In Table 2.1 it can be seen that there are three different ways to obtain the

Algorithm 2.1 A Dynamic Programming algorithm for CSG [RJ08]

```
1:  function DP_CSG(N, u)
2:      for all n ∈ N do
3:          f1[{n}] := {n}
4:          f2[{n}] := u({n})
5:      for s := 2 to |N| do
6:          for all {C|C ⊆ N, |C| = s} do
7:              f2[C] := max_{C'⊂C, 1≤|C'|≤½|C|} f2[C'] + f2[C − C']
8:              if f2[C] ≥ u(C) then
9:                  f1[C] := {C', C − C'}
10:             else
11:                 f1[C] := C
12:                 f2[C] := u(C)
13:     return GET({N})
14:
15: function GET(N)
16:     if |f1[C]| > 1 then
17:         CS = ∅
18:         for all C ∈ f1[C] do
19:             CS := CS ∪ GET(C)
20:     else
21:         CS := {C}
22:     return CS
```

maximum value of 150: {{1}, {2, 3, 4}} ,{{2}, {1, 3, 4}} and {{1 2}, {3, 4}}. This can be related to the fact that there are three different paths from the grand coalition to the optimal CS. This means that redundant calculations are being made. In the improved version they lower this redundancy by avoiding edges that lead to the same node. They only consider a split of a coalition of size $s'' + s'$ into two coalitions of sizes s'' and s', with $s'' \geq s'$ if:

$$s'' \leq n - (s' + s'') \textbf{ or } n = s' + s'' \tag{2.12}$$

Thus only if $n = 3$, will a split into two coalitions of sizes 1 and 2 be considered. This lowers the amount of checks as can be seen in the resulting CSG graph (see Figure 2.6). It does not remove all the redundant checks, as there are still a lot of edges going towards the 4th level, but it does clear up a large amount. More importantly, all the nodes in the CSG graph are still reachable. In [RJ08] they prove that this holds for any number of players.

2.4.3. APPROXIMATION ALGORITHM

In attempt to obtain a more feasible way of generating coalition structures, an approximation algorithm was constructed in [San+99]. This approximation is tightly bound by the amount of agents a and is acquired by solely looking at the

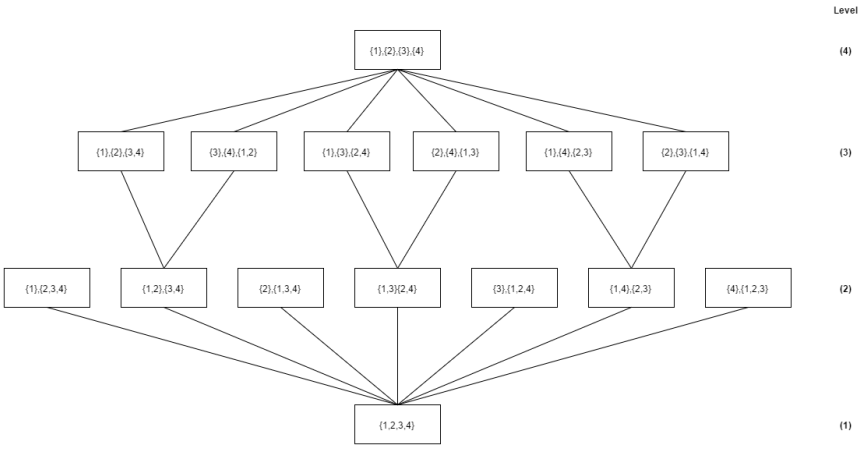


Figure 2.6: Coalition Structure Graph for a game with 4 players while only considering splits according to te rule of [RJ08].

cases where one coalition is split from the grand coalition (see level 2 in Figure 2.4). The resulting coalition structure would then be a set of two coalitions. Since there are 2^{a-1} different coalitions that can split of from the grand coalition, it is required to evaluate at least 2^{a-1} structures. If after doing this, there is still time left, the result can be improved by looking at the next level (coalition structure with three coalitions). Moreover, they show that no algorithm can achieve a bound on the optimal solution with any less then 2^{a-1} evaluations.

Algorithm 2.2 A bounded anytime approximation algorithm for CSG [San+99]

1: **function** DP_CSG(u)
2: Search the bottom two levels of the CSGraph
3: Continue with a breadth-first search from the top of the graph as long as there is time left, or until the entire graph has been searched
4: Return the coalition structure with the highest welfare among the ones seen so far

The problem with this approximation is, that only a small amount of coalitions will be formed. The resulting coalition structure will therefore either have very few equally large coalitions, or at least one very large coalition. Both of these possibilities are far from ideal for the idea of microgrids, where the goal is to partition the grid into many small coalitions. Even though it is possible that there are a few well-formed microgrids, there will always be a bad coalition left. Which provides a lot of unfairness towards the prosumer in the large coalition.

Since this algorithm simply evaluates all coalition structures one by one, it can return the best found so far at any time. So yes, this one does have an anytime property.

The ability to conform to graphical structures uses the same argumentation

as for the previous algorithm. As long as we can find a way to filter out illegal combinations, we can simply evaluate the left over possibilities.

For finding a solution, the coalition structures are checked one by one. Using the previous solution as starting point would mean that you would skip all the coalition structures that come before that one. This would only be useful if the value of the skipped structures did not change in the mean time. Otherwise, there still is the possibility that a better coalition structure exists within the skipped set.

2.5. COALITION STRUCTURE GENERATION OVER GRAPHS

So far, we have seen a number of approaches to generating coalition structures. In general it is assumed that there are no restrictions on who can form a coalition with whom. However, as we have shown in Section 2.2.1, not every combination of players is feasible. Coalitions can be constrained by communication, trust or the ability to share resources.

As Myerson [Mye77] first noted, such constraints can be modelled as a network, where coalitions are only considered feasible if all members are connected with each other, either directly or indirectly through other members of the coalition. In Section 2.2.1 we have seen how this works for the formation of microgrids: a microgrid (coalition) should not be formed if not all members are connected with each other. Disconnected prosumers of the same microgrid cannot share energy with each other, which can lead to unexpected results during isolation. Another example is given by Hoefer, Váz, and Wagner [HVW14]: a situation where scientists collaborate on papers and projects. A scientist does not know every other scientist in the world, and hence can only cooperate with the ones that he does know. These acquaintances span a network where a collaboration of scientists can only exist if they know each other directly or through someone else in the collaboration.

Graph Coalition Structure Generation (GCSG): *Given a graph* $G = (V, E)$ *and a utility function* $u(X)$. *Find a partitioning* $P = \{P_1, \ldots, P_m\}$ *of* V *such that the induced subgraphs of* P *are connected and* $\sum_{P_j \in P}(|P_j| * u(P_j))$ *is maximized.*

Even with the presence of constraints for coalitions, the traditional approaches for CSG can be used. By giving infeasible coalitions a utility of $-\infty$, these coalitions are avoided in order to maximize social welfare. However, this does not mean infeasible coalitions do not participate in the calculation of a solution. As a result, redundant work is being done and calculations take a longer time than they should.

To counter the redundant work, proposals have been made that focus on the graphical constraints of coalitions. Most of this work concentrates on enumerating the possible coalitions (or structures) in a smart way such that infeasible coalitions are not considered. The optimal coalition structure can then be found much faster within this bounded search space.

2.5.1. EXACT ALGORITHM

The most straightforward solution for CSG over graphs is most likely that of Voice, Ramchurn, and Jennings [VRJ12]. They propose a method called SlyCE (Sequentially connected Coalition Enumeration), which can be used to enumerate all feasible coalitions in a graph. On top of SlyCE they propose DyCE (Dynamic Programming for optimal connected Coalition structure Evaluation), an algorithm based on IDP, the algorithm of Rahwan and Jennings [RJ08] (see Section 2.4.2). DyCE improves upon this work by only evaluating the coalitions provided by SlyCE. Though this creates an overhead on the runtime for complete graphs, DyCE speeds up when the density of the graph lowers whereas IDP remains constant.

SlyCE (Algorithm 2.3) is a recursive algorithm that will enumerate all possible coalitions of at most size m, given a current coalition R and expansion possibilities F. In doing so it traverses a tree representation of the set of feasible coalitions in a graph $G = (N, E)$.

Algorithm 2.3 Algorithm for enumerating feasible coalitions in a graph [VRJ12]

```
1: function SlyCE(R, F, m, E)
2:     C := ∅
3:     if F ≠ ∅ and m > 0 then
4:         for all F* ⊆ F with 1 ≤ |F*| ≤ m do
5:             R' := F* ∪ R
6:             F := N̄(F*, R)
7:             C := C ∪ SlyCE(R', F', m − |F*|)
8:     return C
```

Blindly following this procedure will lead to coalitions being redundantly enumerated, as a coalition $\{A, B\}$ can be obtained by starting at $\{A\}$ as well as $\{B\}$. For this reason, F is generated using

$$\overline{N}(F, R) = \{j \mid j > \min(R \cup F)\} \cup N(F) - N(R) \cup R \cup F$$

Where $N(\cdot)$ denotes the set of neighbours of a subset in G (i.e $N(R) = \{j | \exists i \in R, (i, j) \in E\}$). As a result, only those players with a higher id than the lowest in R will be added. This removes the redundancy in the enumeration but still ensures its completeness. Obtaining the complete set will be achieved by running SlyCE$(\emptyset, \{i\}, |N|, E)$ for all $i \in N$.

As mentioned before, the enumeration of SlyCE is used by DyCE to avoid infeasible coalitions. If we compare Algorithm 2.4 to Algorithm 2.1, we can see a few minor differences. First of all f_2 is set to $-\infty$ for all $C \subset N$, f_2 is then overwritten by $u(C)$ for each feasible coalition C found by SlyCE. This initialisation is used to skip infeasible solutions at line 14. Next, SlyCE is used again to obtain the to be evaluated splits at line 15, making sure no infeasible split is considered. The final difference is at line 10-12, where the limit of the split sizes is set, though this is more due to the improvement of Rahwan and Jennings on Algorithm 2.1 than due to that of DyCE.

Algorithm 2.4 DyCE algorithm

```
1:  function DyCE(G = (N, E), u)
2:      for all C ⊂ N do
3:          f_2[C] := −∞
4:      for all i ∈ N do
5:          A :=SlyCE(∅, {i}, |N|, E)
6:          for all C ∈ A do
7:              f_1[C] := C
8:              f_2[C] := u(C)
9:      for all s = 1 to |N| do
10:         m := ⌊s⌋/2
11:         if s < |N| then
12:             m := min(m, n − s)
13:         for all {C|C ⊆ N, |C| = s} do
14:             if f_2[C] > −∞ then
15:                 A := ⋃_{i∈C} SlyCE(∅, {i}, m, {(i, j)|i, j ∈ C, (i, j) ∈ E})
16:                 max := max_{C′∈A} f_2[C′] + f_2[C − C′]
17:                 if max ≥ u(C) then
18:                     f_1[C] := {C′, C − C′}
19:                     f_2[C] := max
20:     return GET(N)
```

The combination of SlyCE and DyCE improves the runtime for CSG over graphs when the graphs are sparse, the worst case runtime, however, does not improve. An improvement for this runtime was found for a special class of CSG instances.

Definition 2.5. *As defined by Diestel [Die05], a* tree decomposition *of a graph $G = (V, E)$ is a mapping of G into a pair (X, T), where $X = \{X_1, \ldots, X_n\}$ with $\forall X_i \in X$, $X \subseteq V$ and T is a tree with X as its nodes such that:*

1. $\bigcup_{X_i \in X} = V$
2. $\forall (v, w) \in E, \exists X_i \in X$ *where* $\{v, w\} \subseteq X_i$
3. *Given two nodes X_i and X_j, $\forall X_k$ on the path from X_i to X_j, $X_i \cap X_j \subseteq X_k$*

In [VPJ12], Voice, Polukarov, and Jennings prove that, using tree decompositions, the runtime of CSG over graphs is bounded by $O(w^{w+O(1)}n)$, where w is the treewidth[5] and n the number of players. Though this only holds for instances where the utility functions are *independent of disconnected members* (IDM).

[5] The treewidth of a tree decomposition of a graph is an indication of how tree-like the graph is. A tree has a treewidth of 1 and a complete graph has a treewidth of $n - 1$.

Definition 2.6. *For a graph $G = (N, E)$, a utility function $u(X)$ is IDM if $\forall i, j \in N$ with $(i, j) \notin E$ and coalition C with $i, j \notin C$,*

$$u(C \cup \{i\}) - u(C) = u(C \cup \{i, j\}) - u(C \cup \{j\})$$

As with the general version of CSG, there also exists an anytime algorithm for CSG over graphs as proposed by Bistaffa, Cerquides, and Rodriguez-aguilar [BCR14]. Allowing for an approximative solution which will be improved while there is still time left.

2.5.2. ANYTIME ALGORITHM

Similar to the usage of SlyCE and DyCE, the anytime algorithm proposed by Bistaffa, Cerquides, and Rodriguez-aguilar [BCR14] starts by generating a search space. Though, instead of generating feasible coalitions, it generates feasible coalition structures. Using a branch and bound algorithm, a search is executed in order to find the optimal coalition structure while limiting the search space.

The search space of coalition structures is constructed using the contraction of edges. Given a graph $G(N, E)$ an edge $(i, j) \in E$ can be contracted to form a new node k, any edge to or from i and/or j is then redirected to k. If we consider each node in G to be a coalition, we start out with a coalition structure of singletons. The contraction of an edge (i, j) then represents the merger of coalitions i and j.

From the starting point of a coalition structure of singletons, the decision of which edge to contract can be seen as a branching decision. The possible decisions then form a search tree with on each node a feasible coalition structure. Whenever an edge is contracted, it is marked. Marked edges are no longer considered for contraction in order to avoid redundant coalition structures. An example of such a search tree can be seen in Figure 2.7.

As we know, the search space of coalition structures is very large. Iterating over all possibilities is therefore not an option. Bistaffa, Cerquides, and Rodriguez-aguilar manage to bound their search space using a special property of their domain: the sum of utility for an entire coalition structure is given by an $m+a$ function ($f = f^+ + f^-$), which is partly monotonic and anti-monotonic[6]. In the context of coalition formation, a utility function is monotonic if for any two distinct coalitions $A, B \subset N$, $u(A \cup B) \geq u(A) + u(B)$ and anti-monotonic when $u(A \cup B) \leq u(A) + u(B)$.

Given a position in the search tree CS, we can conclude that for the anti-monotonic part, no merge of coalitions will improve the solution. At the same time, for the monotonic part, applying every possible merge (by contracting all unmarked edges) will result in a CS^{merge} that maximizes the solution. Let T^{CS} be the subtree of CS, the upper bound of every coalition structure in T^{CS} is given by $UB(T^{CS}) = f^+(CS^{merge}) + f^(CS)$. This upper bound, in combination with the best found solution so far, is used to decide whether a part of the search tree should be investigated or not.

[6] A function g is monotonic (resp. anti-monotonic) if, for all x and y such that $x \leq y$, one has that $g(x) \leq g(y)$ (resp. $g(x) \geq g(y)$). [BCR14]

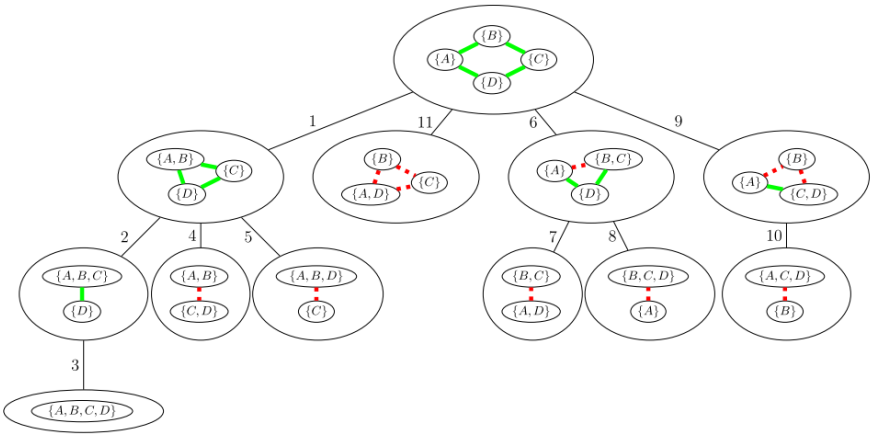


Figure 2.7: A search tree for coalition structures for a square graph. Green edges are available for contraction, red ones are marked and have already been used.

If the algorithm is allowed to finish, an exact solution will be obtained. However, the runtime can also be limited. During the traversal of the search tree, the best found coalition structure so far is stored. Whenever there is not enough time for a complete execution, the algorithm can be stopped to return the best found solution so far.

There certainly are a number of approaches to CSG in graphs. Though, in these approaches an important assumption being made. Which is that the players that participate in the formation of coalitions actually want to maximize the social welfare. While maximizing social welfare is a nice goal for the whole community, each individual can have a different agenda. When players are rational, they only care about the benefits they obtain. A coalition structure that maximizes social welfare could conflict with the interests a number of individuals, which could interfere with its formation as players could refuse cooperation.

2.6. STABILITY OF COALITIONS

Rational players are always looking to maximize their own utility. This makes it so that these players will not stay in their designated coalitions when they can find a better one as there is no restriction that stops a player from leaving a coalition if he wants to. A set of coalitions where no player is inclined to leave is called *stable*.

However, *when* a player is able to "find a better coalition" is loosely defined and has multiple interpretations. In the literature, one can find multiple notions of stability. Probably the most important one being defined by Bogomolnaia and Jackson [BJ02]: *core stability*. This notion of stability is focussed on the possibility of a coalition that improves the utility for all its participants.

Example. *Consider Figure 2.1, where each prosumer v has an energy consump-*

tion/production r_v (indicated by the numbers inside the nodes) and let $u(X) = -|\sum_{v \in X} r_v|$. Let the current set of coalitions be $\{\{A,B,C,D,E,G,H\},\{J,K,L,O\},\{F,N,M\}\}$. Calculating the utility values, we get $u(\{A,B,C,D,E,G,H\}) = -2$, $u(\{J,K,L,O\}) = -1$ and $u(\{F,N,M\}) = 0$. In this situation, for some of the members of the first coalition, there exists a better coalition: $\{D,E,G,H\}$ with $u(\{A,B,C,D,E,G,H\}) < u(\{D,E,G,H\}) = 0$. The current set of coalitions is therefore not considered core stable.

Definition 2.7. *Given a set of players N and partitioning $P = \{P_1,\ldots,P_m\}$ of N. For an $i \in N$, $P(i)$ denotes the coalition $P \in P$ such that $i \in P_j$.*

Definition 2.8. *A partitioning P of a set of players N is* core stable *if $\nexists X \subseteq N$ such that $\forall i \in X$, $u(X) > u(P(i))$.*

Core stability is the most general notion of stability as it considers all available options. However, there also exists a stricter notion: *individual stability*. In contrast to core stability, this notion focusses on the movement of a single player. Here, a coalition structure is considered stable when there is no player that can move to a different coalition, improving his utility and not reducing the utility for the other players of that coalition. Determining the existence of a core stable coalition structure has been shown to be $\sum_2^P$-complete[7] by Woeginger [Woe13] for the case where every player i has its own utility function $u_i(X)$.

Example. *In 2.1, let $P = \{\{A,B\},\{C,D,G,H,J\},\{M,F,N\},\{E\},\{K\},\{L\},\{O\}\}$. As $u(\{A,B\}) = -4$, $u(\{E\}) = -8$ and $u(\{A,B,E\}) = -4$, E can improve its utility by joining $\{A,B\}$. This move is possible since $u(\{A,B\}) \leq u(\{A,B,E\})$. Thus P is not considered individually stable.*

It can also be concluded that P is not core stable but not for the same reasons. As the coalition $\{A,B,E\}$ does not improve the utility for A and B, this is not the coalition that causes the instability. Instead the coalition $\{D,E,G,H\}$ is the coalition possibility that causes instability.

Definition 2.9. *A partitioning P of a set of players N is* individually stable *if there is no $i \in N$ and $P_j \in P \cup \{\emptyset\}$ such that $u(P_j \cup \{i\}) > u(P(i))$ and $u(P_j \cup \{i\}) \geq u(P_j)$.*

As individual stability only concerns the options for a single player, it is a easier to verify than core stability: for every player, you can check whether there is an existing coalition P_j that satisfies the requirements within polynomial time. Individual stability is therefore an easier problem, shown to be NP-Complete by Ballester [Bal04]. Though, it can be said that this notion is also less interesting as it only shows short-sighted decisions of players.

2.7. RESEARCH QUESTIONS

In this chapter we have seen some problems that are close to the formation of microgrids, both for the goal of maximizing the benefit for the entire network (GCSG) as for its individuals (stability). For both problems it has been shown

[7] $\sum_2^P$-complete is also known as NP^{NP}-complete

that neither can be solved within polynomial time. However, for the formation of microgrids it would be ideal to solve the combination of these two problems: maximizing the social welfare while keeping the participants content.

Stable Graph Partitioning (SGP): *Given a graph $G = (V, E)$ and an evaluation function $u(X)$. Find a partitioning $P = \{P_1, \ldots, P_m\}$ of V such that P is core stable, the induced subgraphs of P are connected and the social welfare, $\sum_{j=1}^{m}(|P_j|u(P_j))$, is maximized.*

Because GCSG is already NP-Complete, the addition of an extra constraint, for which verification is also NP-Complete, can only be expected to make the problem more difficult. Furthermore, no work has been found for this particular problem, which also makes it more difficult to find an appropriate technique. The work for CSG and GCSG is therefore the closest we will get towards an applicable existing approach, though adjustments would be needed in order to guarantee stability of solutions. We therefore think it is wise to disregard the constraint of stability for now and start with GCSG first.

In the literature we have seen a number of approaches to GCSG and CSG. First of all, the dynamic programming algorithm of Rahwan and Jennings [RJ08] is an all purpose solution that can take care of restrictions by giving infeasible coalitions $-\infty$ utility. The anytime approach of Sandholm et al. [San+99], on the other hand, is more of a theoretical contribution than a practical one as it simply iterates over every coalition structure using a smart order.

Of the algorithms for GCSG, the SlyCE and DyCE approach of Voice, Ramchurn, and Jennings [VRJ12] is probably the only one that can be applied for microgrid formation as the utility function of this problem does not have the IDM property nor is it an $m + a$ function, which is required for the algorithm of [BCR14].

A short discussion as to why the utility function does not have these properties: in Section 2.2.2 it was stated that microgrids are preferred to have a balanced energy consumption/production, which was given by the $u_b(X)$ function. This means that adding a player to a coalition can result in an increase of utility, when the energy balance approaches 0, as well as a decrease of utility, when the energy balance distances from 0. The utility function for microgrid formation will therefore neither be monotonic or anti-monotonic. The same holds for the IDM property, following a simple example for three nodes A, B and C, where $r_A = -4$, $r_B = 2$ and $r_C = 4$, it can be seen that $u(\{A\} \cup \{B\}) - u(\{A\}) = -1 + 3 \neq -3 + -1 = u(\{A\} \cup \{B, C\}) - u(\{A\} \cup \{C\})$.

As the SlyCE and DyCE algorithm outperforms the IDP algorithm of Rahwan and Jennings for sparse graphs, it can be considered the state of the art approach of GCSG. But even this approach is not able to handle more than tens of nodes (ranging from 30 to 60). The scalability of GCSG is certainly a problem, especially since the power grid is an environment with millions of entities.

For the remainder of this thesis we will need to ensure that SGP is indeed more complex than GCSG as this would an alteration of the constraint of core stability. Next we would need to find a proper substitution for core stability.

Then we can try to find a heuristic for the new version of SGP. In short this thesis will focus on the following research questions:

For the remainder of the thesis we need to determine whether SGP is indeed of a higher complexity than GCSG. If it is, a revision of SGP might be needed in order to make the problem somewhat easier. In our search for suitable techniques for microgrid formation we will therefore answer the following research question:

Is SGP of a higher complexity than GCSG?

In case SGP is too hard to sufficiently solve, a relaxation of the problem will be needed. As the optimization of social welfare will remain the main goal of the problem, the requirement of core stability is probably the only place where SGP can be simplified. As the contentment of participants is important to the problem of microgrid formation, any relaxation of the problem should still conform to the main idea of stability. Under the assumption that SGP with core stability is too difficult, another research question will have to be answered:

What relaxation of SGP is easier to solve, but still provides some form of stability?

Given that GCSG by itself is NP-Complete, a scalable approach to SGP will have to be a heuristic. So in the end, regardless of whether a relaxation is needed or not, a heuristic will have to be found for SGP. This leads to our final research question:

What heuristic can be used to solve SGP(-relaxed) in a scalable manner?

Bondi [Bon00] describes several types of scalability. In this thesis we refer to his definition of *space/time-scalability* when scalability is mentioned as this definition is more focused on the algorithm and its ability to handle increasingly larger problems.

Definition 2.10. *A system is* scalable *if the data structures and algorithms used to implement it are conducive to smooth and speedy operation whether the system is of moderate size or large.*

This definition is dependant on the context as smooth and speedy is a very subjective manner to describe a process. In our context, considering that the power grid is a very large network and we do not know how much larger it will grow, we can only only allow a linear runtime of $O(n)$.

In the remaining chapters of this thesis we will answer these questions one by one and thus provide a solution to our original problem on how to form microgrids.

3

STABLE GRAPH PARTITIONING

In the previous chapter Graph Coalition Structure Generation (GCSG) was described: a problem closely related to microgrid formation. This problem was used in the formulation of our own extended version: Stable Graph Partitioning (SGP). Thus far, a suitable approach for SGP has not yet been found and on top of that SGP is expected to be too difficult to solve. In this chapter we will prove that SGP is indeed harder than GCSG. This will lead to the definition of a new notion of stability, which will make SGP easier. In order to guarantee this new notion of stability, the existing DyCE algorithm will be altered and a new distributed algorithm will be constructed.

3.1. COMPLEXITY OF SGP

Even though the problem of deciding the existence of a core stable solution was proved to be $\sum_2^P$-Complete[1] by Woeginger, the complexity of SGP could be different as every player has the same utility function in SGP.

We can prove that a core stable solution always exists when the utility function is uniform for all players. This would mean that there is a special class of problem instances for which determining the existence of a core stable solution is solvable in polynomial time: check whether the utility function is uniform for all players, if so return true.

Theorem 3.1. *Given a set of players V, if $\forall i, j \in V, u_i(X) = u_j(X)$, then a core stable partitioning always exists.*

The proof that a core stable partitioning exists for these cases will follow by proving that an algorithm, that always returns a core stable solution in these circumstances, is correct.

Algorithm 3.1 is such an algorithm. Given a set of players V and a utility function $u(X)$ it finds an $X \subseteq V$ with that maximizes $u(X)$, X is then added to

[1] Also known as NP^{NP}-Complete

the final partition P and its players are removed from V. These steps are repeated until $V = \emptyset$, after which P contains all the coalitions that form a core stable partitioning of V.

Algorithm 3.1 Algorithm that finds a core stable partitioning of a set of players with a uniform utility function

```
1: function CORE(V, u(X))
2:     P := ∅
3:     while V' ≠ ∅ do
4:         Let X ⊆ V such that u(X) = max_{C⊆V} u(C)
5:         V := V − X
6:         P := P ⋃ {X}
7:     return P
```

Lemma 3.2. *Given a set of players V, and a utility function $u(X)$ that is uniform for all players of V. Algorithm 3.1 will return a core stable partitioning.*

Proof. Given that Algorithm 3.1 returns a partitioning $P = \{X_1, \ldots, X_m\}$, P will be proven to be core stable. This proof is done inductively by showing that a subset of V is core stable given the partitioning P. That is, for increasing l:

$$\nexists X \subseteq V, \text{ such that } \bigcup_{i=1}^{l} X_i \cap X \neq \emptyset \text{ and } \forall v \in X, u(X) > u(P(v)) \tag{3.1}$$

When $l = m$ we obtain $\bigcup_{i=1}^{l} X_i = V$, after which (3.1) becomes the regular notion of core stability[2].
Basis: $l = 1$: As a result of Algorithm 3.1, for $P = \{X_1, \ldots, X_m\}$, X_1 is a coalition such that $u(X_1) = \max_{C \subseteq V} u(C)$. Therefore $\nexists X \subseteq V$ with $u(X) > u(X_1)$ and thus (3.1) holds.
Induction hypothesis: Suppose (3.1) holds for all l up to some k, $k \geq 1$.
Induction: Assume (3.1) holds for $l = k$. Assume (3.1) does not hold for $l = k+1$. Then there should be an $X \subseteq V$, such that $\bigcup_{i=1}^{k+1} X_i \cap X \neq \emptyset$ and $\forall v \in X, u(X) > u(P(v))$. As (3.1) holds for $l = k$, $\bigcup_{i=1}^{k} X_i \cap X = \emptyset$ and therefore $X_{k+1} \cap X \neq \emptyset$. It follows that $X \subseteq V - \bigcup_{i=1}^{k} X_i$ and $u(X) > u(X_{k+1})$. Algorithm 3.1 chooses X_{k+1} such that $u(X_{k+1}) = \max_{C \subseteq V - \bigcup_{i=1}^{k} X_i} u(C)$ thus there can be no $X \subseteq V - \bigcup_{i=1}^{k} X_i$ with $u(X) > u(X_{k+1})$. This is a contradiction, thus (3.1) holds for $l = k+1$.

By induction it follows that Lemma 3.2 is indeed true. □

By proving Lemma 3.2, we have shown that there exists a method that will find a core stable partitioning for a set of players for the case where $\forall i, j \in V, u_i(X) = u_j(X)$. Indirectly, we have proven that there exists a core stable solution for this class of instances, thus Theorem 3.1 also holds. The complexity of core stability in SGP is clearly different in contrast to what was originally proven and this could possibly influence the complexity of SGP itself.

[2] It always holds for an $X \subseteq V$, $X \neq \emptyset$ that $X \cap V \neq \emptyset$.

In Section 2.7, two problem definitions were formulated, one for the dynamic version of the problem and one for the static. We can claim that the original SGP problem is at least as hard as its static version as a reduction from the static version to SGP can be made: create a random partitioning P' from V and take $u(P', X) = u(X)$. But, for the sake of determining the appropriate complexity class, we will alter the problem one step further by transforming it into a decision problem.

SGP-Decide: *Given a graph $G = (V, E)$, an evaluation function $u(X)$ and an integer k. Is there a partitioning $P = \{P_1, \ldots, P_m\}$ of V such that P is stable, the induced subgraphs of P are connected and $\sum_{i=1}^{m} u(P_i) \geq k$.*

It is known that an optimization problem is always at least as hard as its decision counterpart, making it a useful tool to determine the lower bounding complexity of the optimization problem. We will provide some arguments as to why SGP-Decide $\in \sum_2^P$ and possibly $\sum_2^P$-Complete, though we do miss a few important results to indefinitely prove this. Our argumentation is as follows:

- First we show that SGP-Decide $\in$ NP-Hard and that a subproblem, *Stability*, is Co-NP-Complete.
- This result leads to the conclusion that SGP-Decide $\notin$ NP $\cup$ Co-NP
- Finally we show that SGP-Decide $\in \sum_2^P$ (though not $\sum_2^P$-complete)

Lemma 3.3. *SGP-Decide $\in$ NP-Hard*

Proof. We can show that SGP-Decide $\in$ NP-Hard by reducing an NP-Complete problem to it within polynomial time. For this purpose we will use the Partition problem, which is already known to be NP-Complete.

Partition problem: *Given a set of integers $I = \{i_1, \ldots, i_n\}$, can I be split into two subsets I_1 and I_2, such that $\sum_{i \in I_1} i = \sum_{j \in I_2} j$?*

Let $s = \sum_{i \in I} i$, then we can create a reduction from Partition to SGP-Decide as follows:

1. $\forall i \in I$ create a vertex $v \in V$ with $R_v = i$ and $u(X) = -|\frac{1}{2}s - \sum_{w \in X} R_w|$
2. Let $G = (V, E)$ be a complete graph.
3. Take $k = 0$.

The reduction is valid iff for every YES-instance of Partition, the reduction will result in a YES-instance of SGP-Decide and for every instance of Partition that results in a YES-instance of SGP-Decide, the instance is a YES-instance of Partition. Some notes upfront: considering the evaluation function $u(X) = -|\frac{1}{2}s - \sum_{v \in X} R_v|$, the maximum value is 0[3]. $u(X) = 0$ iff $\sum_{v \in X} R_v = \frac{1}{2}s$. Due to the reduction, $\forall v \in V$, $R_v \in I$. Thus, the maximum value of $u(X)$ can only be obtained if there exists an $I_1 \subset I$ with $\sum_{i \in I_1} i = \frac{1}{2}s$.

[3]The maximum of a function $f(x) = -|x|$ is $f(0) = 0$.

Now assume there is a valid partitioning of I. Then there exist two subsets I_1 and I_2 such that $\sum_{i\in I_1} i = \sum_{j\in I_2} j$. In other words $\sum_{i\in I_1} i = \frac{1}{2}s$. This means that there is a $P = \{P_1, P_2\}$ that partitions V such that $u(P_1) = u(P_2) = 0$, fulfilling $\sum_{i=1}^{m} u(P_i) \geq k = 0$. Since G is a complete graph, the induced subgraphs of P are connected and since $u(P_1) = u(P_2) = 0$ is the maximum obtainable utility, $\nexists X$, such that $\forall w \in X,\ w \in P_w \in P,\ u(X) > u(P_w)$, meaning P is also stable. Thus the reduction of a YES-instance of Partition is a YES-instance of SGP-Decide.

Next, assume the reduction of an instance of Partition is a YES-instance of SGP-Decide. Then there is a $P = \{P_1, P_2\}$ that partitions V such that $\sum_{i=1}^{m} u(P_i) \geq k = 0$. Since $\max\{u(X)\} = 0$, $u(P_1) = u(P_2) = 0$, therefore $\sum_{v\in P_1} R_v = \sum_{v\in P_2} R_v = \frac{1}{2}s$. From the reduction follows that $\bigcup_{v\in V} R_v = I$, thus I can be split I_1 and I_2, such that $\sum_{i\in I_1} i = \sum_{j\in I_2} j$. Therefore an instance of Partition that is reduced to a YES-instance of SGP-Decide, is a YES-instance of Partition.

Hence there exists a valid reduction within polynomial time from Partition to SGP-Decide and as a result SGP-Decide $\in$ NP-Hard. □

Noticeable in our proof is that the requirement of stability in SGP-Decide does not influence the possibility to reduce Partition to SGP-Decide. Merely the requirement of utility maximization would have sufficed. However, the stability requirement is a difficult part of the problem nonetheless. This fact hints at the possibility of SGP-Decide being a harder problem than Partition. Therefore, we also consider the complexity of *Stability*.

Stability: *Given a partitioning* $P = \{P_1, \ldots, P_m\}$ *of* V *and an evaluation function* $u(X)$. *Does it hold for* P *that* $\nexists X \subseteq V$ *such that* $\forall i \in X\ u(X) > u(P(i))$?

Lemma 3.4. *Stability* $\in$ *Co-NP-Complete*

Proof. Stability $\in$ Co-NP-Complete iff Stability $\in$ Co-NP and Stability $\in$ Co-NP-Hard. We will start by showing that Stability $\in$ Co-NP. Thus, for a given instance $I = (P, u(X))$ and a certificate X, it needs to be verifiable within polynomial time that I is a no-instance.

Algorithm 3.2 Verification that an instance $I = (P, u(X))$ is a no-instance using a certificate X.

```
1: Let X ⊆ V such that ∀i ∈ X, u(X) > u(P(i))
2: for all i ∈ X do
3:     if u(X) ≤ u(P(i)) then
4:         return UNDECIDED
5: return NO-INSTANCE
```

As X is the given certificate it does not influence the runtime, furthermore there are at most n elements in X and finding the corresponding set $P(i)$ is also doable in $O(n)$ time. Thus, with algorithm 3.2, it has been shown that verification of a no-instance is possible within polynomial time.

In order to prove that Stability $\in$ Co-NP-Hard, we will show that *Tautology*, a Co-NP-Complete problem, is reducible to Stability.

Tautology: *Given a boolean formula f containing the variables $B = \{b_1, \ldots, b_n\}$, is f satisfied for each assignment of B?*

In our reduction, we consider that an $X \subseteq B$ represents an assignment where $\forall b_i \in X$, b_i is true and $\forall b_i \notin X$, b_i is false. As an empty coalition cannot exist, an additional variable c will be added to represent the case where all $b_i \in B$ are set to false. The reduction can be created as follows:

1. Take $P = \{B \cup \{c\}\}$, where c is an additional variable.
2. Take $u(X) = \begin{cases} 1 & \text{if } (c \notin X \text{ and } !(X \text{ satisfies } f)) \text{ or } (\{c\} = X \text{ and } !(\emptyset \text{ satisfies } f)) \\ 0 & \text{otherwise} \end{cases}$

First of all, it is trivial that this reduction can be executed in polynomial time.

The reduction is valid iff for every YES-instance of Tautology, the reduction will result in a YES-instance of Stability and for every instance of Tautology that results in a YES-instance of Stability, the instance is a YES-instance of Tautology.

Assume f is a tautology, then there exists no assignment such that f is not satisfied. As $P = \{B \cup \{c\}\}$ there is only one coalition and $u(B \cup \{c\}) = 0$. An improving coalition can only be found when there is an assignment that does not satisfy f. Since no such assignment exists, there is no improving coalition and thus P is stable. Thus the reduction of a YES-instance of Tautology is a YES-instance of Stability.

Assume P is stable, then there is no X such that $u(X) = 1$. As of such, there is no assignment for which f is not satisfied. Thus f is a tautology. Therefore an instance of Tautology that is reduced to a YES-instance of Stability, is a YES-instance of Tautology.

Thus there exists a valid reduction in polynomial time from tautology to Stability and therefore Stability $\in$ Co-NP-Hard.

Since Stability $\in$ Co-NP-Hard and Stability $\in$ Co-NP, Stability $\in$ Co-NP-Complete. □

With SGP-Decide being NP-Hard and its subproblem, Stability, being Co-NP-Complete, we are pointed in a direction where SGP-Decide might be outside of NP and Co-NP. The complexity class we then reach is a subclass of the polynomial hierarchy: $\sum_2^P$. This is a class for which a YES-instance is verifiable within polynomial time with the use of an *NP-Oracle*, a black box program that can decide an NP-problem within polynomial time.

Theorem 3.5. *SGP-Decide* $\in \sum_2^P$

Proof. We can show that SGP-Decide $\in \sum_2^P$ by providing an algorithm that can verify a YES-instance of SGP-Decide in polynomial time using an *NP-Oracle* (see algorithm 3.3).

Even though this problem uses an Oracle for the Co-NP-Complete problem Stability, it is still considered an NP-Oracle. A Co-NP problem is the complement of an NP-Problem, meaning that a decider of a Co-NP problem can return the complement of the result of the corresponding NP decider. Thus, Co-NP and NP-Oracles are equal in complexity. □

Algorithm 3.3 Verification that an SGP-Decide instance $I = (P, u(X))$ is a NO-instance.

```
1: function VERIFYSGP-DECIDE(G = (V,E), u(X), k)
2:     Let P partition V.
3:     STABILITYORACLE(P, u(X))
4:     sum := 0
5:     for all P_i ∈ P do
6:         ISCONNECTED(P_i)
7:         sum := sum + u(P_i) * |P_i|
8:     return sum ≥ k
```

As of yet, we are unable to show that SGP-Decide $\in \sum_2^P$-Complete. The complexity proof of Woeginger [Woe13] relied heavily on a set of players with different utility functions, so that proof is not reusable for SGP-Decide. While giving every player the same utility function had the effect of making the search for a core stable solution easier, the addition of a minimum social welfare could make the problem harder again. However, since both stability and social welfare heavily depend on the same utility function, it is hard to find a proper reduction from existing $\sum_2^P$-Complete problems.

Nonetheless, we have shown that SGP is indeed more complex than the NP-Complete problem GCSG. While core stability is a nice property to have in microgrid formation, it simply is too difficult. We should therefore first focus on a solution for GCSG and then determine what kind of stability can be achieved.

3.2. A NEW NOTION OF STABILITY

The main reason core stability is so difficult is the large amount of possibilities for the coalitions. This is mainly due to an important aspect of core stability: everyone has perfect knowledge of the entire environment. However, in practice obtaining perfect knowledge is not very likely nor should it be needed. The further away two players are situated, the less likely it is that they will form a coalition. If it is unlikely that two players will ever form a coalition, why would they need information about each other? It seems more intuitive to only have knowledge of those players that are in your neighbourhood.

The size of a neighbourhood can be determined by the maximum number of hops one can take to reach a "neighbour". In a complete graph of n nodes, a neighbourhood with 1 hop will contain all n nodes. But, in sparser graphs with an average degree d, this neighbourhood size will shrink to d nodes. In particular, a neighbourhood with k hops contains $O(d^k)$ nodes (note that this number cannot exceed n).

Definition 3.1. *Let $d(i,j)$ be the length of the shortest path between i and j. Given a graph $G = (V,E)$ and an integer k, the k-neighbourhood of an $i \in V$ is defined as $N_k(i) = \{j \mid j \in V, d(i,j) \leq k\}$*

When players are aware of fewer other players, their possibilities also de-

crease. With less possibilities, the problem of stability becomes easier to solve. An added addition is that the average size of a neighbourhood will not increase when more nodes are added to the graph as long as the average degree stays the same.

Using the definition of a k-neighbourhood, a new notion of stability can be defined: k-stability. With this notion of stability, it is assumed that no one is aware of any players outside of their neighbourhood. Players are therefore unaware of any possible coalitions that are not contained within their neighbourhood. As of such, a coalition that provides at least as much utility as the best coalition within the neighbourhood of its members can be considered a stable coalition.

Definition 3.2. *Given a graph $G = (V, E)$, a partitioning P of V is* k-stable *if* $\forall i \in V \nexists X \subseteq N_k(i)$ *with* $u(X) > u(P(i))$.

3.2.1. THE EFFECT OF K-STABILITY

Using k-stability instead of core stability for SGP will have an impact on the obtained solutions depending on the chosen value of k. This impact is mainly focused on the stability and social welfare of the solution. When k increases, players have more knowledge of the network. It is expected that as a result, partitions will become more stable.

Theorem 3.6. *Given a partition P that is $(k+1)$-stable, then P is also k-stable.*

Proof. From the definition of k-stability, we know that for P it holds that $\forall i \in V \nexists X \subseteq N_{k+1}(i)$ such that $u(X) > u(P(i))$. By definition $N_k(i) \subseteq N_{k+1}(i) \forall i \in V$. Thus for P it holds that $\forall i \in V \nexists X \subseteq N_k(i)$ such that $u(X) > u(P(i))$. Therefore P is k-stable. □

The higher k, the more stable the partition. On the other hand, it is expected that a more stable solution will result in more restrictions for usable coalitions and therefore a lower social welfare. Intuitively it holds that the more selfish players are, the lower social welfare will be.

Theorem 3.7. *Given the maximum social welfare W_k for an SGP instance with k-stability, $W_k \geq W_{k+1}$.*

Proof. In order to obtain a W_{k+1}, there should be a P such that P is $(k+1)$-stable and $\sum_{P_i \in P} |P_i| u(P_i) = W_{k+1}$. Following Theorem 3.6 it holds that P is k-stable. Thus $W_k \geq \sum_{P_i \in P} |P_i| u(P_i)$, therefore W_k is at least as high as W_{k+1}.

We can show that it is also possible for W_k to be higher than W_{k+1} with the example of Figure 3.1. Given a utility function $u(X) = -|\sum_{i \in X} r_i|$, the coalition with the highest individual utility is that of {A, B, C, D} with $u(\{A, B, C, D\}) = -5$. As this is the coalition with the highest individual utility, a 2-stable solution would be P ={{A, B, C, D}, {E}} with $\sum_{P_i \in P} |P_i| u(P_i) = -31$. Disregarding stability, the solution for maximum social welfare would be P_{max} ={{A, B, C, D, E}} which would result in a social welfare of −30. No one is aware of {A, B, C, D}

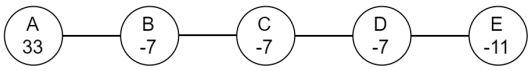


Figure 3.1: An example graph where, if the utility function were $u(X) = -|\sum_{i \in X} r_i|$, a 1-stable solution can obtain a higher social welfare than a 2-stable solution.

when $k = 1$, thus P_{max} is 1-stable. When $k = 2$ however, {A, B, C, D} is within the neighbourhood of B and C, thus P_{max} is not 2-stable.

As W_k can never be lower than W_{k+1}, but it can be higher, we can conclude $W_k \geq W_{k+1}$. □

Social welfare and stability counteract each other. With higher values of k more stability is achieved, but the social welfare worsens. The decision of k therefore not only influences the difficulty of the problem, but also the resulting solutions.

3.2.2. THE VALUE OF k

So far, the effect of the value of k has been discussed, but it is still quite intangible what this value truly is. From the description of k-stability, we know that k defines the maximum amount of hops between two neighbours. k, in combination with the topology of a graph $G = (V, E)$, defines the neighbourhood $N_k(i)$ $\forall i \in V$. It is a given that the neighbourhood of any $i \in V$ cannot be larger than V. Thus increasing k beyond a point where $\forall i \in V\ N_k(i) = V$, has no effect at all.

For each graph there exists some bound for which it is meaningless to have k exceed that. When for every player the neighbourhood contains the entire network, there is no longer a reason to increase k as the neighbourhood cannot grow any larger anyway. At this point k-stability will have become equivalent to core stability since every player will have complete knowledge of the network.

The value of k for which core stability is achieved is dependant on the topology of the network. In a complete graph, the k-neighbourhood of each player already covers the entire graph when $k = 1$. Graphs where players are further apart, on the other hand, will require higher values of k.

The dependency on topology would suggest that the *average degree* of a graph dictates the maximum value of k. But this is far from the truth.

Definition 3.3. *The* average degree *of a graph $G = (V, E)$ reflects the average number of edges each vertex has and is given by $\frac{2|E|}{|V|}$.*

Graphs with the same average degree (and the same amount of vertices) can have a very different bound for k. This behaviour can be seen in the next example.

Example. *Given a linear graph $G_{lin} = (V_{lin}, E_{lin})$ where $V_{lin} = \{v_1, \ldots, v_n\}$ and $E_{lin} = \{\{v_1, v_2\}, \{v_2, v_3\} \ldots, \{v_{n-1}, v_n\}\}$ and a star graph $G_{star} = (V_{star}, E_{star})$ where $V_{star} = \{v_1, \ldots, v_n\}$ and $E_{star} = \{\{v_1, v_2\}, \{v_1, v_3\} \ldots, \{v_1, v_n\}\}$. As $|V_{lin}| = |V_{star}|$ and $|E_{lin}| = |E_{star}|$, the graphs have the same average degree. However, for G_{star} the neighbourhood of every player contains the entire graph when $k \geq 2$, while for G_{lin}, this only occurs when $k \geq n - 1$.*

Under some circumstances, even sparse networks can have large neighbourhoods with a relative small value of k. Sparsity does not seem to be the deciding factor for the neighbourhood size. The major difference between G_{star} and G_{lin} is that every vertex in G_{star} can reach every other vertex within 2 steps, while for G_{lin} this amount is much higher. This attribute is called the *diameter*.

Definition 3.4. *The* diameter *of a graph $d(G)$ is the maximum length of the shortest path between any pair of vertices.*

We can show that the required value of k for k-stability ≡ core stability is upper bounded by the diameter of a graph.

Theorem 3.8. *Given a graph $G = (V, E)$, k-stability is equivalent to core stability when $k \geq d(G)$*

Proof. The effect of $d(G)$ is that for each $i \in V$ every other vertex is reachable within $d(G)$ hops. In other words, $N_{d(G)}(i) = V \ \forall i \in V$. By definition k-stability is equivalent to core stability when $N_k(i) = V$ for all $i \in V$. Thus k-stability is equivalent to core stability when $k \geq d(G)$. □

With the upper bound for k it can be concluded that for any given graph $G = (V, E)$, k should be an integer in the range of 0 to $d(G)$. However, this does not mean for lower values of k core stability is unachievable. It depends on the problem instances. If a supposedly core stable partition solely consists out of singleton coalitions then core stability can even be achieved with $k \geq 0$. An empirical study will have to decide how k-stability will behave for our practical problem of microgrid formation.

3.3. CENTRALIZED ALGORITHM

k-stability has been defined as one of the constraints of SGP, but we do not yet have a way of find k-stable solutions. The existing DyCE algorithm is able to maximize social welfare but without considering stability. In the previous section it was mentioned that the best choice for an individual is not necessarily the best choice for the community, but the opposite can also be said: the best choice for the community is not necessarily the best choice for an individual. Maximizing social welfare does not ensure k-stability. Thus DyCE needs to be altered to conform to the additional requirement of k-stability.

Forming a k-stable partition works kind of the same way as that of core stability: by partitioning V using coalitions with high values of utility (see Lemma 3.2). For core stability this meant that the best coalition in a set of players V would be chosen, but for k-stability, the chosen coalitions only need to be better than what is available in each player's neighbourhood.

Algorithm 3.4 DyCE algorithm altered to provide k-hood stable solutions.

1: **function** SDyCE($G = (N, E), u, k$)
2: **for all** $C \subset N$ **do**
3: $f_2[C] := -\infty$
4: **for all** $i \in N$ **do**
5: $A :=$SLYCE($\emptyset, \{i\}, |N|, E$)
6: **for all** $C \in A$ **do**
7: $f_1[C] := C$
8: $f_2[C] := u(C)$
9: $f_3[C] :=$ FINDMAX(C, k)
10: $f_4[C] := u(C)$
11: **for all** $s = 1$ to $|N|$ **do**
12: $m := \lfloor s \rfloor / 2$
13: **if** $s < |N|$ **then**
14: $m := \min(m, n - s)$
15: **for all** $\{C | C \subseteq N, |C| = s\}$ **do**
16: **if** $f_2[C] > -\infty$ **then**
17: $A := \bigcup_{i \in C}$ SLYCE$(\emptyset, \{i\}, m, \{(i, j) | i, j \in C, (i, j) \in E\})$
18: $max := \max_{C' \in A} f_2[C'] + f_2[C - C']$
19: $bestC := \max\{f_4[C'], f_4[C - C']\}$
20: **if** $bestC \geq f_3[C]$ AND $max \geq u(C)$ **then**
21: $f_1[C] := \{C', C - C'\}$
22: $f_2[C] := max$
23: $f_4[C] := bestC$
24: **return** GET(N)
25:
26: **function** FINDMAX(C, k)
27: $max := -\infty$
28: **for all** $i \in C$ **do**
29: $max := \max\{max, f_3[C - \{i\}]\}$
30: **if** $\exists i \in C, C \subseteq N_k(i)$ **then**
31: $max := \max\{max, u(C)\}$
32: **return** max

In the DyCE algorithm, given a $C \subseteq V$, information was already stored on the most beneficial split (f_1) and the resulting utility (f_2). For our alteration, we need to keep track of two extra things:

1. The best possible coalition available in the neighbourhood of players within C (f_3).

2. The best coalition of those C will split into (f_4).

For a $C \subseteq V$ should only be split into C' and $C - C'$, when either $f_4[C'] \geq f_3[C]$ or $f_4[C - C'] \geq f_3[C]$. Applying these changes results in Algorithm 3.4.

The FindMax function of Algorithm 3.4 is quite straightforward except for the $\exists i \in C,\ C \subseteq N_k(i)$ expression. However, this can be solved by calculating the shortest paths of the subgraph $G' = (C, E)$ using a Floyd-Warshall algorithm and verifying whether there is a path longer than $2k$. If the path is longer, then C cannot be within the neighbourhood of a single player.

As $\forall i \in V\ N_k(i) \subseteq V$, the available coalitions will typically have lower utility than those possible in the entire set. A solution with k-stability therefore has more freedom than that of a core stable one.

A centralized solution, has a number of quirks. It lacks the ability to handle larger number of players in an efficient way. Existing solutions have only be shown to be able solve to problems with a maximum of 60 players. This is a strong indication that a centralized algorithm will not be able to form microgrids in the power grid, where the number of players (households) can easily exceed a million.

Furthermore, the production and consumption of energy is constantly changing. Gathering and processing the information of over a million entities in one central point will also provide problems.

Given the complexity of the problem (see Section 3.1) and the size of practical problems, a centralized approach is not going to be the final solution. A more promising approach would be distributed, where coalitions can be formed from the bottom up.

3.4. DISTRIBUTED ALGORITHM

It is common knowledge that a distributed algorithm cannot cover for the size explosion of the problem. This is understood as the growth from the network solving the problem cannot keep up with the growth of the problem, just like a supercomputer would not be able to keep up with its increase of processors. In order to be useful for practical cases of SGP, the envisioned distributed algorithm will therefore be a heuristic. As we have not found any work on this particular problem, evaluation of such a heuristic will occur in relation to the centralized algorithm.

The main idea of the distributed algorithm is to let every prosumer in the grid think for himself. This means that every vertex in the graph, will be a distinct processor that can calculate options and communicate with the rest. We will assume that the network only restricts which coalitions can be formed, not the

communication between nodes, meaning that any two nodes can communicate with each other regardless of their position in the network.

Communication occurs with the use of messages. To indicate what kind of information a message contains, different types will be used. At the same time, a number of states will be used to determine what actions should be taken and which messages should be processed. What the different states and message types are, will be explained at a later point in this section.

3.4.1. BASIC APPROACH

The basic approach fo the distributed algorithm is an approach where the coalitions will only expand. Individual players will each start out as a singleton (a coalition with only one member). In order to improve their utility, each coalition will gather *information* on the other coalitions situated within its neighbourhood (defined by k) and evaluate possible merges.

Out of the possible merges, a coalition chooses the optimal one[4] and formulates that merger into a *proposal*. Proposals are exchanged between neighbours after which each coalition can compare received proposals with their own. This allows a coalition to consider merges that involve coalitions outside of its own neighbourhood. Any received proposal that does not contain the coalition in question is not considered as an option, but is used as a notification in order to determine whether a coalition should wait for more proposals or continue.

Out of all received proposals (including its own) each coalition chooses the one that provides the most improvement for its utility. Once this proposal has been chosen, a *reply* is send to each proposer in order to inform them of this decision.

If a proposer receives a positive reply of each participant of the proposal, it can consider the proposal to be accepted. If there are one or more negative replies, a proposal can be considered rejected. Once a proposer has determined the status of a proposal, a *confirmation* will be send to all positive repliers to inform them whether the merge will take place or not. If a proposal has been accepted, the participating coalitions will form a new coalition, otherwise nothing will happen. Either way, the coalition will restart by gathering information on its neighbourhood.

A summary of this process can be seen in Figure 3.2. It is repeated until no merges occur in consecutive rounds.

Even though coalition can exist out of multiple players, they make decisions and take actions as a single entity. This is possible since the utility function is the same for everyone, hence there is no real need for discussion within a coalition. To realize this behaviour, every coalition has a leader randomly elected in order to make decision for the coalition.

During the execution of the distributed algorithm, there are two important decisions being made: which coalition is to be proposed and which of the proposals is going to be pursued. As these decisions depict in what way the coalitions will grow, they have a large influence on the performance of the algorithm.

[4] The merge that provides the most gain in utility.

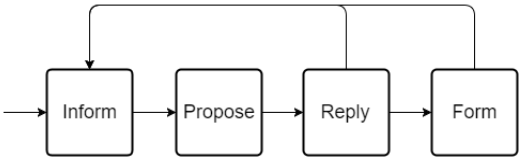


Figure 3.2: The flow of a node in the distributed algorithm. Starting by gathering information, proposing a better coalition, replying to received proposals and forming a new coalition if a proposal was accepted. These steps are repeated until no new coalitions are formed.

3.4.2. PROPOSAL CREATION

Given a coalition C and its neighbouring coalitions $N_k(C)$, C is looking for the best new coalition X with $C \in X$ and $X - \{C\} \subseteq N_k(C)$. Finding the best coalition within the neighbourhood will be done in a similar matter as with the DyCE algorithm: by enumerating and evaluating the options using the SlyCE algorithm. For coalitions with an equal amount of utility, a random decision is made.

Normally, iterating over all combinations would seem like a bad idea, given that there can be as many as d^k distinct neighbouring coalitions and thus 2^{d^k} possibilities. However, a study on the characteristics of the power grid, conducted in [PA12] observed that the average degree of the power grid network is somewhere around 3. This makes us believe that we can get away with a brute force approach as long as k is sufficiently low[5].

A downside to this approach is that proposals are blindly made by pursuing the best coalition possible. However, there can be occasions where one or more of the participants of a proposal keep on rejecting it, either because they are content with their current coalition or because they are pursuing other proposals themselves. In such a situation, the proposer is not making any progress since it keeps on proposing something that will be rejected anyway.

Whenever a coalition rejects a proposal, it is unlikely that it will accept that same proposal the next round. Additionally, since the proposal usually contains the best thing the proposer can come up with, any modifications to it will likely be rejected again by the rejecter. A coalition therefore maintains a list of rejecters. Any coalition in this list will be temporarily disregarded in the creation of subsequent proposals. The result is a new proposal which will possibly provide less utility, but will be more likely to be accepted.

In case of the rejecter forming a new coalition, it will be removed from the list as this change opens up new possibilities and therefore a new chance for a proposal to be accepted. The same goes for when the proposer forms a new coalition, but in this case the entire list is discarded. Finally, when the rejected list contains the entire neighbourhood, the list is reset.

After the creation and transmission of proposals, each coalition will have to determine what to reply. For each network that does not represent a complete graph, the neighbourhoods of coalitions can differ. As of such, the pro-

[5] Whether it is actually doable or not the brute force over the combinations will possibly be shown in the experiments.

posed coalitions can be different, even though each proposer initially believes that their proposal is the best one possible. An evaluation of received proposals is therefore still needed in order for a coalition to determine whether it should pursue its own proposed coalition or that of one of his neighbours.

3.4.3. PROPOSAL EVALUATION

The evaluation of received proposals is more straightforward than their creation. A coalition C has $|N_k(C)| = m$ neighbours, which means that C will receive at most m proposals that include itself. For each of these proposals, the one that results in the highest amount of utility will be chosen. For proposals with equal utility an additional calculation is required.

A random decision for proposals with equal utility is not preferred as it could lead to bad results.

Example. *If a coalition B proposes $\{A,B,C\}$ and C proposes a coalition of equal utility $\{B,C,D\}$, there is a possibility that B and C do not chose the same coalition. As a result, neither of these coalitions will come into existence even though there is no good reason as to why B and C should not be able to come to an agreement.*

In order to get two parties to agree to the same proposal, the random factor should be removed. Thus instead of deciding randomly, a deciding factor is used that is the same for each coalition that evaluates a proposal.

Each proposal contains a coalition that is being proposed. When comparing two proposals with respectively a coalition C_1 and C_2 where $u(C_1) = u(C_2)$, we find for both coalitions the player with the smallest id that is not in the other coalition:

$$id_{C_1} = \min\{i | i \in (C_1 - C_2)\}$$
$$id_{C_2} = \min\{i | i \in (C_2 - C_1)\}$$

The coalition for which this id is the smallest is then chosen as the "better" proposal.

With the definition of proposal creation and evaluation, the main functionality of the algorithm has been described. But there is still one aspect that needs to be discussed: the algorithms greedy growth.

3.4.4. GREEDY GROWTH

The greedy growth of the distributed algorithm is an easy way to find coalitions fast. On the other hand, it also gets the algorithm stuck in some local optimum fast. Due to the algorithms' incapability to search around a local optimum, the result is highly dependent on early made choices.

Example. *The effect of the greedy growth can be seen in Figure 3.3. It is the coalition structure formed as a result of the simplistic algorithm. The decision of creating the coalition $\{G,J,K,O\}$ has removed the possibility for D to find a coalition for himself, with four singleton coalitions as a result.*

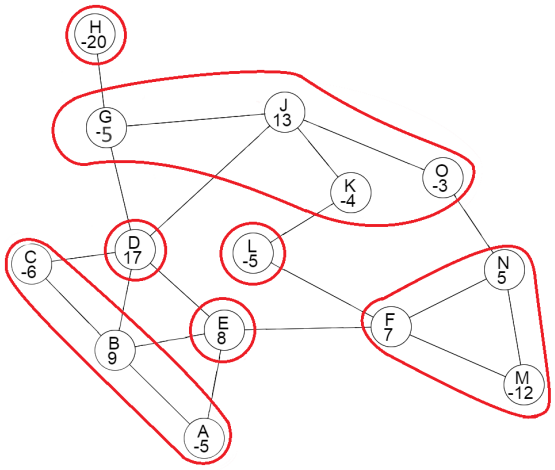


Figure 3.3: A coalition structure formed by the simplistic distributed algorithm.

If we were to make a slight adjustment by removing G and adding L, the coalition $\{D,G\}$ *could come into existence which would then expand to* $\{D,E,G,H\}$. *This coalition structure would provide a lot more social welfare.*

To counter this weakness, some extra work should be added once a local optimum is reached. Because it is always a coalition that blocks the possibilities of smaller coalitions, the larger coalition should operate as a mediator. During this state with coalition structure P, each coalition $C \in P$ should check if it can improve a neighbouring coalition $B \in P$ by excluding one of its members $i \in C$. At the same time, this coalition can try to find a replacement R for the to-be-excluded-member (if none is available $R = \emptyset$ with $u(R) = 0$. The resulting operation should then only be executed if it improves the social welfare.

$$u(C) + u(B) + u(R) < u(C - \{i\} \cup R) + u(B \cup \{i\})$$

The improvement of social welfare is required in order to prevent operations that deter the solution. An example would be the operation of switching C to $\{D\}$ as $u(\{C,D\}) > u(\{D\})$ as this would not help the solution in any way.

With the new distributed algorithm we have obtained a heuristic that provides a k-stable partitioning of a network while approximating the maximum social welfare. However, it remains a question whether this algorithm is scalable and how useful k-stability is as opposed to core stability.

4

EXPERIMENTAL EVALUATION

In the previous chapter it was shown that Stable Graph Partitioning (SGP) is probably harder than Graph Coalition Structure Generation (GCSG). Consequentially, a relaxation of core stability, k-stability, was proposed in order to simplify SGP. In Section 3.2 some characteristics of k-stability for different values of k were proven: the equivalence to core stability when k is equal to the diameter of the network and the effect of a low k as opposed to a high one. However, no recommendation for the most suitable value of k was given. Setting k to equal the diameter of the network would be ill-advised as this would make the stability problem as hard as core stability, whose complexity was the reason k-stability was created in the first place. Without a decent suggestion for the value of k, the k-stability is not quite usable as a relaxation for the problem.

To improve the likelihood of obtaining a scalable algorithm for SGP, the chosen value of k will have to be significantly smaller than the diameter of large networks. However, k cannot be arbitrarily small as it should still come close to the idea of core stability. In order to determine the appropriate k, the following question will have to be answered:

Q. 1 *What is the most fitting value of k for k-stability?*

The most fitting value will have to be small to sufficiently reduce the search area for coalitions, but large enough for k-stability to be representative of core stability. If such a value can be found, k-stability would be proven to be a useful relaxation of core stability as it would reduce the amount of work needed to find proper coalition structures and at the same time maintain the concept of stability.

We hypothesize that k-stability can approach core stability even when k is small. Coalitions receive less utility the larger they are (see Section 2.2.2), thus it is likely that optimal coalitions will be relatively small. This means that optimal coalitions are likely to be found within neighbourhoods that cover only a small part of the network. The fact that larger, less optimal coalitions cannot be found

with the use of small k should not conflict with the notion of core stability where the focus is to find the best individual coalition anyway.

H. 1 A small k is the most fitting value for k-stability.

Admittedly, "small" is subjective, but the experiments will provide an exact value. In our opinion a small k is significantly smaller than the diameter of a network. This would certainly reduce the amount of work necessary to find k-stable coalitions, although whether this reduction is enough for the algorithm, constructed in Section 3.4, to be scalable for this k remains unknown.

The value of k determines the size of the neighbourhood and thus how fast the difficulty increases as the problem size increases. For neighbourhoods with larger k, it is more likely that an additional node will fit in that neighbourhood as they cover a larger portion of the network than neighbourhoods with smaller k. If every additional node will be part of more neighbourhoods, the average neighbourhood size will grow faster. Thus the growth of neighbourhood sizes will likely be slower for smaller k.

Q. 2 *Is the distributed algorithm scalable for small values of k?*

Considering the scalability of the algorithm, it is hypothesized to rely on the value of k. As was shown in Theorem 3.8, there is a point where k-stability becomes equivalent to core stability. It was already proven that SGP with core stability is too hard to solve, even for a heuristic, so at this point it is very unlikely that the distributed algorithm is scalable. In order to narrow this down, we should look at the growth of neighbourhood size for a given k.

Recall Figure 3.2, in our algorithm, every node i iterates over four distinct phases: inform, propose, reply and form. Of these phases, the propose phase is the most computational intensive as each node traverses its entire neighbourhood and evaluates each possible subgraph ($O(2^{N_k(i)})$ combinations). Considering that our utility function $u(X)$ is computable within $O(|X|)$, the runtime of each iteration of a node is $O(2^{N_k(i)} N_k(i))$. Given a total of n nodes, a single iteration will be executed n times. As each iteration at least one coalition is formed, there will be $O(n)$ iterations. Let $N_k^{max} = \max_i N_k(i)$, the expected runtime of the algorithm is $O(2^{N_k^{max}} N_k^{max} n^2)$.

In a complete network, the neighbourhood size would be equal to the total amount of nodes and our algorithm would perform in the order of $O(2^n n^3)$. However, we consider power grid networks, which are far from complete. It is hypothesized that in sparse networks the neighbourhood sizes will eventually reach an asymptote (in a sparse network) and that $O(N_k^{max}) \ll O(n)$. The neighbourhood sizes could even be small enough such that $O(2^{N_k^{max}} N_k^{max}) < O(n)$ which would indicate the scalability runtime of the algorithm as $O(n^3)$.

H. 2 The distributed algorithm is scalable when used in combination with small values of k.

The research for the optimal value of k will be commenced using the exact algorithm constructed in Section 3.3, it will be used to confirm the effect of k on

the stability and social welfare of the solutions and to find a threshold for k for which the stability does not improve much further. For the performance of the distributed algorithm, we will test it against against different values of k. The scalability will be determined using the increase of runtime as the problem size increases.

4.1. METHODS

In order to test H. 1 and H. 2, we need to determine what needs to be measured and how. For H. 1 we already deduced that the smaller k is, the better it would be for the reduction of the amount of work. What is left is determining how small k can be in order to provide a sufficient representation of core stability. This means that we somehow need to measure how close an obtained coalition structure is to being core stable. Even though core stable coalition structures could theoretically be obtained by the algorithm as long as k is sufficiently high, it is unlikely that a solution will be given within reasonable time for all problem sizes. For these cases, the obtained k-stable coalition structures can therefore not be directly compared to core stable ones.

As it is impractical to obtain the core stable coalition structures for each problem instance, a direct comparison between k-stable and core stable coalition structures is not possible. Instead of comparing solutions where k is equal to the diameter of the network, a different approach can also be used. From Theorem 3.6 we know that a $(k+1)$-stable solution is at least as stable, if not more, than a k-stable solution. This means that for every increase in k we get closer to core stability. If under these circumstances a k-stable solution is subsequently comparable to a $(k+1)$-stable solution, the stability is either on a plateau or it is core stable.

Measuring the difference in stability can be done in two ways. The first one is to count the number of players in the k-stable solution that have not improved their utility in the $(k+1)$-stable solution and divide that over the total number of players. This will result in a value between 0 and 1 where 1 means that the k-stable solution is also $(k+1)$-stable (the closer to 1, the better). Basically this measures the ratio of $(k+1)$-stable players in the k-stable solution.

The other option is to measure how content unstable players are. This is done by dividing their utility in the k-stable solution by the utility obtained in the $(k+1)$-stable solution. This shows how much worse of the unstable players are due to the usage of a lower k. This measure is called *satisfaction*.

For the testing of H. 2 we have to measure the scalability of the algorithm given a set value of k. Normally in a distributed algorithm, the processing capacity increases automatically when the network size increases. In the experimental environment, this capacity is limited and cannot grow along with the problem size. The algorithm is therefore possibly slower during the experiments than it would be in practice.

On the other hand, the processing capacity will never grow faster than the problem size. While additional capacity would make the algorithm faster, it cannot speed up the algorithm more than the increase in problem size would

slow it down. In other words, if the runtime of an algorithm with a set amount of processing units leads to the conclusion that it is unscalable, it would also be considered unscalable with a dynamic amount. Thus in order to determine whether the algorithm is scalable, the runtime will be measured for different problem sizes and its growth will be compared to that of a linear regression line.

4.2. MATERIAL

Having discussed how the experiments will be conducted, it is now time to discuss what was used for the experiments. This section will explain how the experiments were set up and which hardware was used. Finally the used problem instances were explored.

4.2.1. IMPLEMENTATION

All algorithms were implemented in Java 8. For the graph representation we used a third party library called GraphStream [Bal+10]. This provided us with a basic data structure, tools to generate graphs and a means to visualize created graphs. Furthermore, in order to make our distributed algorithm truly distributable, we used Java's Remote Method Invocation (RMI). This system allows applications to call object methods situated in a separate machine, providing the necessary tools required for message passing between the different nodes.

Experiments were performed on the TU Delft cluster of the DAS-4 [ASC]: a distributed supercomputer. The TU Delft cluster consists out of 32 dual-quad-core compute nodes where each node has a clock speed of 2.4 GHz and 24 GB memory. Twenty of these nodes were used in our experiments by dividing the players of each network over them. Using an extra node, each experiment was set up on the different nodes. This master node was merely used to establish the IP-addresses of each node and record the needed time to obtain a solution, it did not contribute to the computation itself. Furthermore, as the DAS-4 is a shared facility, execution time for each experiment had to be limited to 15 minutes.

4.2.2. PROBLEM INSTANCES

The initial plan for obtaining problem instances was to use existing network topologies like those of the IEEE bus system examples. However, there were only five different topologies with a size ranging from 14 to 300 nodes, so the examples are too few in number and lack large instances. Instead using them for the experiments they were used as a reference point for the creation of new topologies. This is needed as random graphs are unlikely to uphold the characteristics that are apparent in an electricity network. A method that allows for the generation of pseudo-random networks that conform to these characteristics would be much more adequate.

The attributes that we focused on were the average degree of the network and its diameter. These attributes are important for the difficulty of the problem since they dictate the expected neighbourhood size. While the average degree determines the size of a k-neighbourhood for $k = 1$, the diameter determines how fast the neighbourhood sizes grow when k increases. Given two networks of

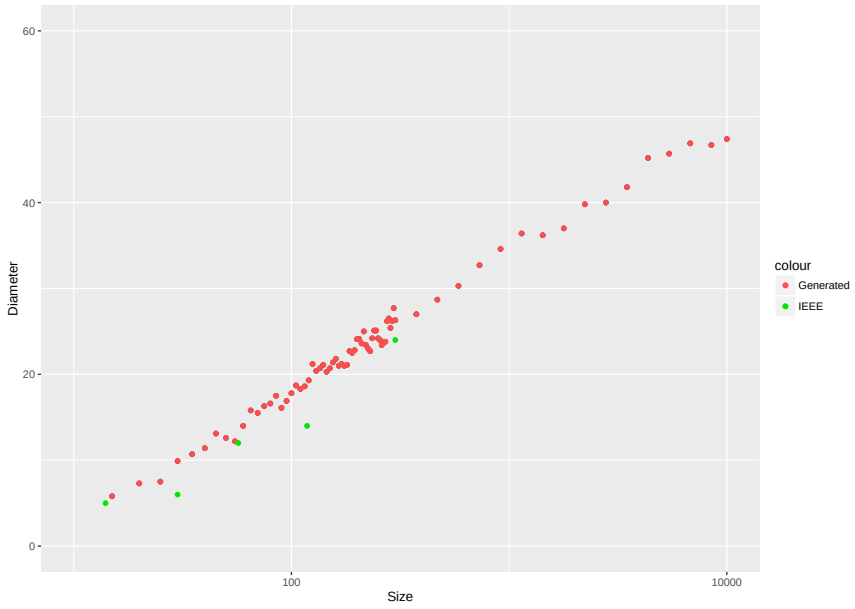


Figure 4.1: The diameter of the generated networks (red) and the IEEE instances (green). If k were to be set to these respective values for each instance, core stability would be achieved.

n nodes but different diameters, for $k > n$, the average neighbourhood of these networks will be the same (namely n), but for the network with a larger diameter, k will have to be larger than for a network with a small diameter before this maximum will be reached.

The average diameter of the IEEE instances was 2.7 and the diameter for each instance can be seen in Figure 4.1.

Our method of topology construction works by first distributing nodes in a 2-dimensional plane and constructing a minimal spanning tree. Then edges are added until the desired average degree is obtained. For the addition of an edge, a random source node is chosen. Next a target node is chosen either at random or by searching for the node, which is not yet a neighbour, that is closest to the source node. The decision for a random target node occurs with a probability p (based on distance occurs with $1 - p$).

As we can precisely choose the amount of edges, the average degree will always match up with the desired value. The diameter of the network on the other hand happens to be tunable with p. For completely random edges ($p = 1$) the network diameter ended up lower and for edges only based on distance the diameter would be too high. With $p = 0.2$ the diameter showed a trend reasonably comparable to that of the IEEE instances (see Figure 4.1).

With this generation method, the problem instances of the experiments were constructed. In steps of 5, networks of 15 to 300 vertices were created. Following that, the size was further increased with a factor 1.5 until a network of 10 000

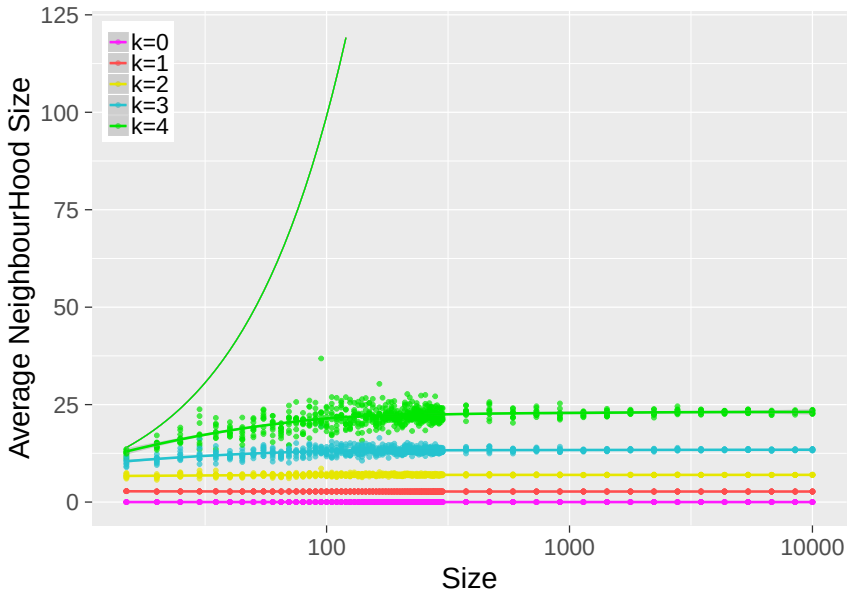


Figure 4.2: The maximum neighbourhood size plotted against the size of the generated problems displayed for various values of k. The top green line represents the maximum neighbourhood size possible (i.e. the network size).

vertices was reached. For each problem size 10 different variations were generated.

The diameter of the generated networks can be seen in Figure 4.1. Following Theorem 3.8, core stability will be achieved if k were to be set to these values for the given problem sizes. This is also the point where the k-neighbourhood of every vertex will cover the whole network.

Our goal is to determine the usability of k-stability for small values of k, so the k will be much smaller than the diameter. As the neighbourhood size is very important for the development of the algorithm, some initial investigation was conducted on the neighbourhood sizes for varying k. The results can be seen in Figure 4.2. It can be seen that for each k the average neighbourhood size converges to a certain value. Even for $k = 4$, the average neighbourhood size stays relatively small. This should be positive for the algorithm as it is likely that the search for optimal coalitions will be conducted in small neighbourhoods.

However, it could also be that a single player will become the bottleneck of the algorithm. A single player with a large neighbourhood will take a long time deciding on his appropriate coalition. In the mean time the other players will have to wait for his decision. That is why, the maximum neighbourhood size was also investigated (see Figure 4.3).

Sadly, the maximum neighbourhood sizes are much larger than the average ones, especially for $k = 4$ and $k = 3$. This could result in the existence of bottle-

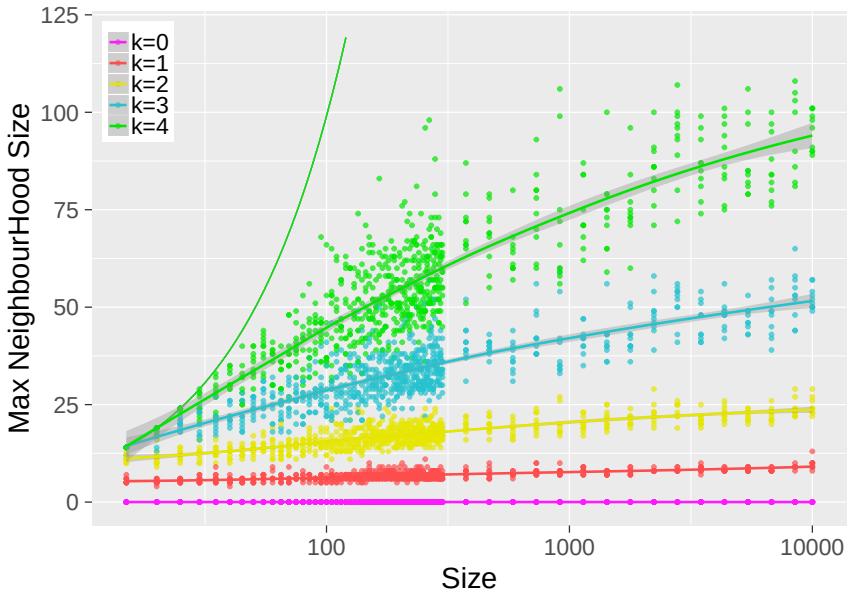


Figure 4.3: The maximum neighbourhood size plotted against the size of the generated problems displayed for various values of k. The top green line is the represents the maximum neighbourhood size possible (i.e. the network size).

necks in the algorithm: the members of these large neighbourhoods will take a very long time to come up with a proposal, making the surrounding nodes wait for a long time. The experiments will show whether the algorithm can overcome these bottlenecks, or that it will result in an unscalable algorithm.

In addition to the structure of te networks, each node was assigned an energy attribute to indicate its production/consumption. As a reminder, this attribute is used in the evaluation function of the coalitions. The amount of energy was random for each node, negative or positive. But, in order to make sure that there is at least a possibility of energy balance, the final energy amounts were adjusted such that the sum of energy in the entire graph was 0.

An additional input parameter for the algorithm is the utility function. This is the function that decides what a beneficial coalition is and what not. For this function there are also variations possible. But, as to not let the number of experiments explode, we have decided to use a single utility function, namely the one that was determined in Section 2.2.2 to reflect the quality of a microgrid:

$$u(X) = 0.2 \cdot \sigma(u_s(X)) + 0.8 \cdot \sigma(u_b(X)) \tag{4.1}$$

4.3. RESULTS & DISCUSSION

As mentioned before, two types of experiments were conducted. One for determining whether the preferred k for k-stability and one for judging the scalability

Size	k	Stability	Satisfaction
15	0	0.8484848	0.9550897
20	0	0.7928571	0.9575348
15	1	0.9393939	0.9689302
20	1	0.9142857	0.9818506
15	2	1.0000000	1.0000000
20	2	1.0000000	1.0000000
15	3	1.0000000	1.0000000
20	3	1.0000000	1.0000000

Table 4.1: Average results for the stability and satisfaction produced by the exact algorithm for SGP.

of the algorithm. For each hypothesis the results will be separately presented and discussed.

4.3.1. THE VALUE OF k

In order to determine the recommended value of k, the centralized algorithm (which produces exact results) was initially used. Sadly the largest problem instances it could handle were those of 20 vertices. The resulting stability and satisfaction can be seen in Table 4.1. Here, the value of $k = 0$ was used additionally to simulate the effect of disregarding stability. Both stability and satisfaction were measured against the results of $k+1$ (e.g. for $k = 1$, stability was measured against the results of $k = 2$). Satisfaction was only measured when $stability < 1$, in order to observe that, whenever there was instability, how much worse off the unstable players were.

From the results of $k = 0$ it is visible that even when coalitions focus solely on social welfare some form of stability can be provided. Furthermore, though around 20% of the network ought to be able to improve their utility, the resulting improvement is on average less than 5%. For $k = 1$ the resulting coalition structures are more stable, however, the available improvement of utility is comparable to the improvement for $k = 0$. Finally, core stability seems to be achieved for $k = 2$, as stability does not improve any more when k is increased. As the coalitions are stable, the players are fully satisfied as well.

This is a positive result, even though it is only for the small problem instances, as it proves that core stability can be achieved with a k that is less than a third of the diameter of the network (see Figure 4.1). Though these results do not show whether this will hold for the larger problems as well.

As these initial results only provide a small indication on the behaviour of k-stability, the investigation needed to be expanded. In order to get more results, the distributed algorithm was also used to compare the stability for various k. This produced slightly different results, as here the social welfare is no longer guaranteed to be maximized. Additionally, for $k = 3$ and $k = 4$, the algorithm eventually went beyond the time limit for the experiments. Thus we were unable to measure the stability for $k = 2$ and $k = 3$ for all instances.

Following the results of Figure 4.4, it can be seen that the stability for $k = 1$,

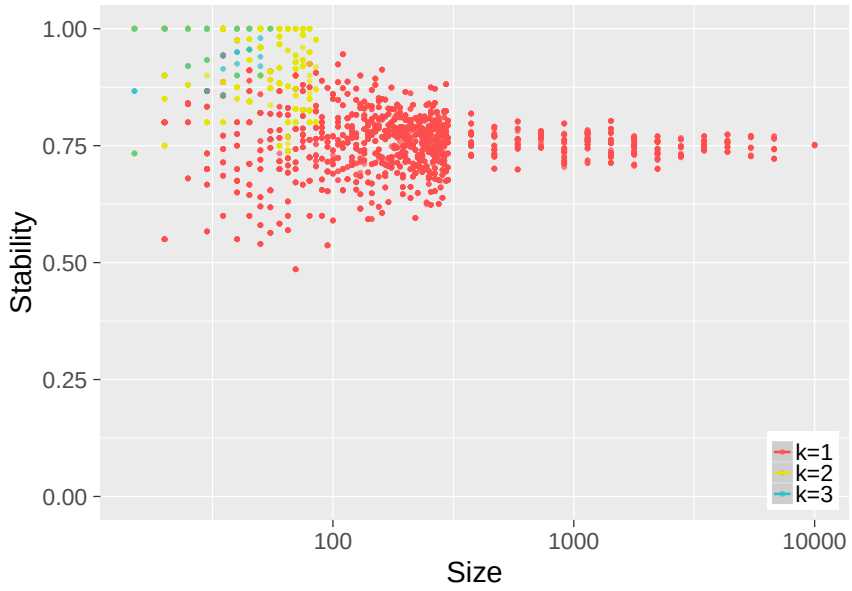


Figure 4.4: The measured stability of obtained coalition structures for various k (relative to the result of $k+1$) compared to the increase in problem size.

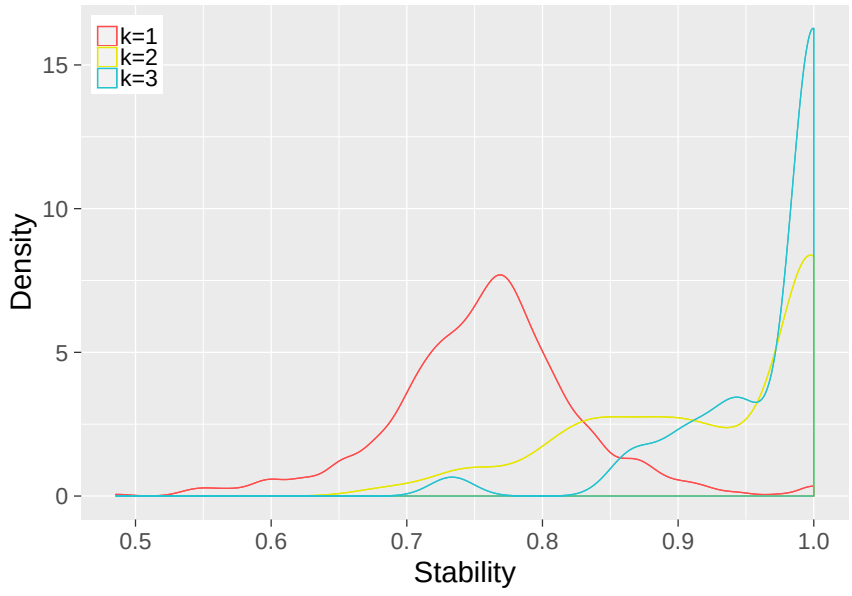


Figure 4.5: A frequency histogram of the values of stability displayed in Figure 4.4. It shows the density of obtained stability values for various k.

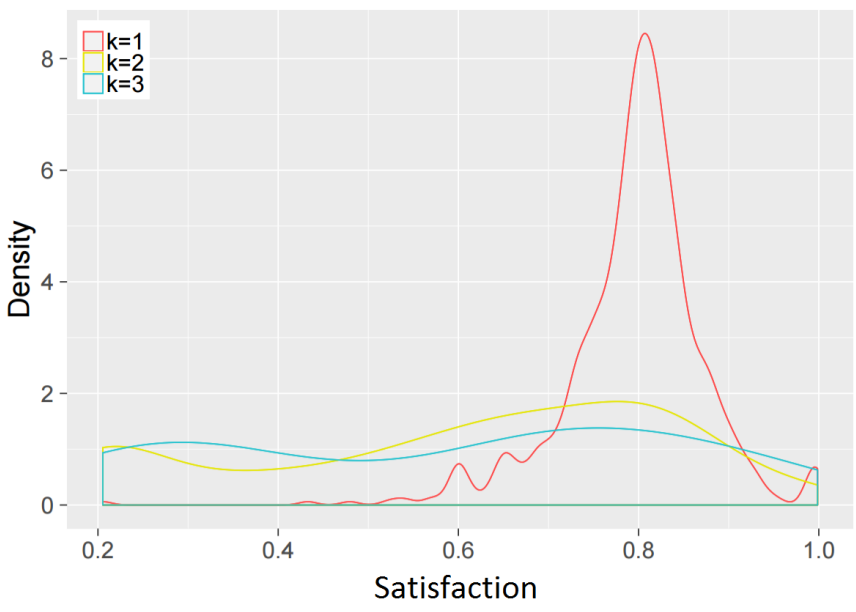


Figure 4.6: A frequency histogram of the satisfaction of found coalition structures for various values of k.

compared to that of $k = 2$ is on average somewhere around 0.75. This means that at around a quarter of the players could increase their utility if k were to be increased to 2. The partitions for $k = 2$ are more stable, as the lowest stability value for this k is at around 0.75. Likewise, the partitions for $k = 3$ are even more stable.

One could also say that the improvement of stability for each increment of k rapidly decreases. This becomes even more evident when the density of the stability values is plotted (see Figure 4.5). Here we see the distribution of the obtained stability values. For $k = 1$ the majority of stability is somewhere around 0.76 whereas for $k = 2$ the majority is situated at around 1.0 and even more so for $k = 3$.

As expected, the stability of the partitions increases as k increases. But, it is remarkable that the increase in stability for each increment in k deteriorates so fast. On the other hand, the satisfaction is a lot worse when k is increased as is shown in Figure 4.6.

Even though the partitions for $k = 1$ are less stable, the unstable players can improve their utility less compared to the more stable partitions for higher values of k. Use of a $k > 1$ ensures that more players obtain a stable coalition, but on the other hand players can end up in a much worse coalition then what should be their stable coalition.

Concerning Q. 1, our hypothesis H. 1 can be accepted. k-Stability with $k = 2$ seems to approach core stability quite closely and is also significantly smaller

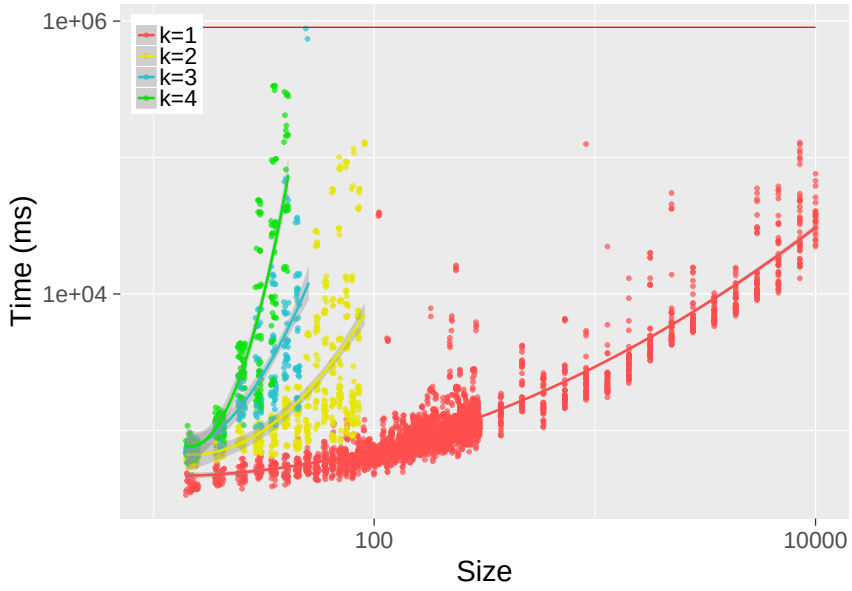


Figure 4.7: The runtime for each problem instance given a certain value of k. Both axis are displayed on a log-scale. Each run on an instance was limited to a maximum of 15 minutes, indicated by the red line.

than the diameter of the networks. $k = 1$ also seems acceptable if a further reduction of neighbourhood size is needed. This choice however, comes at the cost of stability as formed coalition structures are clearly more unstable than for $k = 2$. Besides the need for a small neighbourhood size, one could also motivate this choice with the resulting satisfaction as it was shown that even in unstable coalitions, the obtained utility can only be improved by 5%.

4.3.2. DISTRIBUTED ALGORITHM PERFORMANCE

With k-stability being useful for SGP, even when k is small, the question remains whether the constructed distributed algorithm is scalable. The expectations are that the algorithm is scalable for $k = 1$ and $k = 2$. To this end, the runtime of the distributed algorithm will be evaluated.

The first results we will consider, are those of Figure 4.7 where the algorithm is clearly unscalable for $k = 3$ and $k = 4$. As was stated before, this was expected due to the relatively large neighbourhood sizes. The performance for $k = 1$ also shows nothing out of the ordinary: reasonable trend for the runtime versus problem size.

Remarkable is that for $k = 2$ the algorithm is unexpectedly slow. According to the given neighbourhood sizes, the performed calculations should all be within tolerable sizes. An investigation showed, however, that during the execution of the algorithm some players would end up with a neighbourhood size exceeding

30 players. This is unlike what was shown in Figure 4.3, where the neighbourhood size does not exceed 25 for problem instances of less than 1000 players. While the data on this figure is definitely not untrue, the maximum neighbourhood size does seem to change during the execution of the algorithm.

Consider the formation of a coalition, during this formation, all outgoing edges will be gathered and used in the remainder of the algorithm. This means that the coalition will continue with a union of the neighbourhoods of its members. The aggregation of vertices results in a higher degree of the graph and therefore an increase in neighbourhood sizes. This growth is large enough that the algorithm cannot keep up.

To counter this behaviour and still maintain the k-stability guarantee, an alternative version of the algorithm was used in subsequent experiments. In this version, the neighbourhoods just start out regularly for the given value of k, but once a coalition is formed it will continue as if $k = 1$ in order to limit subsequent neighbourhood sizes. The resulting runtime can be seen in Figure 4.8.

There is no surprise that, for $k = 1$, nothing changes. For $k = 2$ the algorithm runs a lot faster than previously, though there is a lot of variance in the runtime. For the other values of k, the runtime is definitely improved, but it is not enough to counter its rapid increase.

The variance for $k = 2$ can be explained by the differences in neighbourhood sizes. In Figure 4.3 it can be seen that the neighbourhood size varies more as k becomes larger. Given that searching for coalitions inside a given neighbourhood is very difficult, even a small difference in size can result in a large difference in runtime.

For both k, a regression analysis was conducted. The results are shown in Table 4.2. The regression for $k = 2$ provides a different result than $k = 1$. This can be mainly subjected to the variance that is more abundant in the results for $k = 2$. While the exponential model provides a somewhat better fit for the data of $k = 1$ in comparison to the polynomial models, it is not decisive as the fit for the data of $k = 2$ is comparable for both polynomial and exponential models. From the estimated coefficients we can conclude that the correct model is not $O(n^2)$ or $O(n^3)$ as these coefficients are very small or even negative. These values suggest that the coefficients were only used to provide a better fit for the data better and not because the model is of an actual higher degree.

Our initial hypothesis for the scalability of the distributed algorithm for small k was that it would be scalable. From our data we can neither accept or reject this hypothesis as our regression analysis does not provide us with a conclusive model for the runtime. It is not clear whether the model of the algorithm fits a linear one or an exponential one.

The inconclusiveness of the analysis is mainly due to the influence of the neighbourhood size on the runtime. We have determined the runtime to be bounded by $O(2^{N_k^{max}} N_k^{max} n^2)$, but this cannot be modelled to the network size directly. Figure 4.3 gave an indication that $O(N_k^{max}) << O(n)$, but this might not be enough to counter the exponential nature of the runtime.

So it remains a question whether the algorithm is truly scalable. Though we

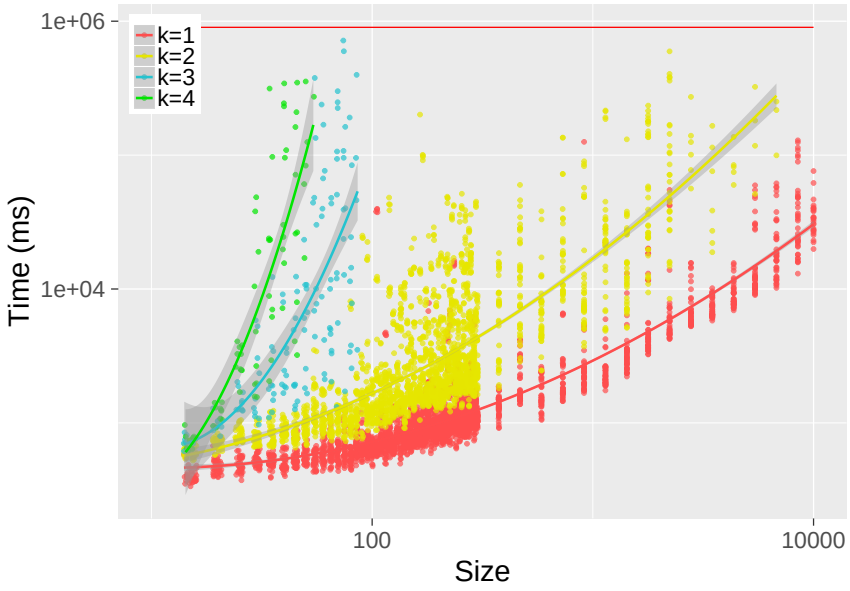


Figure 4.8: The new runtime performance for the adjusted algorithm. Again, both axis are displayed on a log-scale. Here, the k-neighbourhood is only used when a player is not yet in a coalition. Subsequent coalitions are searched in a neighbourhood as if $k = 1$.

can take comfort in the fact that the algorithm was able to find coalition structures in networks of 10 000 nodes within 15 minutes as the exact algorithm of [VRJ12] took 2.6 hours for a network of 40 nodes.

k	Model	R^2	Estimated Formula
1	$O(n)$	0.5768	$Time = 198 + 3.75 \cdot Size$
1	$O(n^2)$	0.5893	$Time = 715 + 2.01 \cdot Size + 0.000218 \cdot Size^2$
1	$O(n^3)$	0.5906	$Time = 937.9 + 0.8923 \cdot Size + 0.0005997 \cdot Size^2 - 2.840 \cdot 10^{-8} \cdot Size^3$
1	$O(e^n)$	0.7409	$Time = e^{6.77+0.00047 \cdot Size}$
2	$O(n)$	0.3938	$Time = -1323 + 32.94 \cdot Size$
2	$O(n^2)$	0.3990	$Time = -2582 + 38.85 \cdot Size - 0.0013 \cdot Size^2$
2	$O(n^3)$	0.3992	$Time = 2879 + 40.64 \cdot Size - 0.0023 \cdot Size^2 + 8.9 \cdot 10^{-8} \cdot Size^3$
2	$O(e^n)$	0.405	$Time = e^{7.54+0.0013 \cdot Size}$

Table 4.2: Regression analysis of the runtime in Figure 4.8 for $k = 1$ and $k = 2$.

5

CONCLUSION

During this thesis a number of results were obtained. In Section 3.1 it was shown that Stable Graph Partitioning (SGP) is likely a complex problem. Though we were unable to prove SGP $\in \sum_2^P$-complete, the combination of the stability requirement and the goal of social welfare maximization suggests that SGP can only be solved by using an NP-oracle. Under the assumption that $P \neq NP$, therefore $SGP \notin NP$ and $SGP \in \sum_2^P$. The requirement of core stability was an important part for the formulation of SGP and could therefore not be disregarded so easily. Therefore the decision was to formulate a relaxation of core stability.

Consequentially k-stability was defined: a new notion of stability where the knowledge of each player is limited to neighbours that are reachable within k hops. It was shown that this notion of stability can be tuned using the value k. It is equivalent to core stability when k equals the diameter of the network and approaches a situation without considering stability when $k = 0$. Furthermore, it was shown that increasing k would increase stability, but decrease the maximum obtainable social welfare. During the experiments in Section 4.3.1 it was determined that even for small values of k, such as $k = 2$, a stability is achieved that is close to that of core stability as subsequent increases of k hardly improved the stability of the solutions. This was an important result as the usage of k-stability in combination with such a small k reduces the search area of individuals significantly.

Finally, a distributed algorithm was constructed in Section 3.4 that could guarantee this form of stability. This algorithm assumes a starting point where everyone is in a singleton coalition. From here, everyone explores possible merges with other coalitions around them (fitting within their k-neighbourhood) and formulates a proposal containing the best merge they found. Proposals are then exchanged after which participants of these proposals reply whether they support this new coalitions or not. Finally, the coalitions of proposals that have full support are formed and the process repeats.

The experiments that focused on determining the scalability of this algo-

rithm were not so successful. A regression analysis of its runtime was inconclusive. We were unable to determine the correct model fitting the obtained data and thus unable to determine whether the algorithm is scalable or not.

This result does not rule out that there is no scalable heuristic for solving SGP with k-stability. As explained in Section 3.4.2, the algorithm uses a brute force approach when searching for optimal coalitions. This leaves room for a more efficient approach, though it will take some effort to maintain the guarantee of k-stability.

During this thesis, three research questions were formulated:

1. Is SGP of a higher complexity than GCSG?

2. What relaxation of SGP is easier to solve, but still provides some form of stability?

3. What heuristic can be used to solve SGP(-relaxed) in a scalable manner?

Comparing the obtained results to these questions, we can conclude that the first two have been appropriately answered. GCSG is known to be NP-Complete whereas SGP $\notin$ NP was proven in this thesis. Therefore, SGP is indeed a more complex problem than GCSG.

Concerning the second question, k-stability has been shown to be an appropriate substitution for core stability in SGP when $k = 2$. In addition, the suggested value for k results in a significant reduction of required work, thus making the problem easier. Though it should be noted that the usage of k-stability does not lead to a lower complexity classification.

The third question is still left unanswered as our proposed heuristic has not been proven to be scalable. While it is clear that the runtime would be exponential for complete networks, the sparsity of power grid networks allows for some leeway.

Though we have shown that the algorithm was able to solve SGP instances of 10 000 nodes within 15 minutes given a limited distributed environment. This is a positive result as the existing work for the similar problem of coalition structure generation, discussed in Section 2.5, indicated to be at its limits with network sizes of around 60 nodes.

In relation to our initial problem this means that we have not entirely solved it. Recall that our initial problem was to find suitable techniques for microgrid formation. We have paved the way by exploring the nature of this problem and proposing a mathematical counterpart that is both solvable and intuitive. This means that we have a clear idea of what the problem of microgrid formation is and how we can simplify it.

5.1. FUTURE WORK

In respect to the problem of microgrid formation, there are a few subjects that were not covered, but which would still be interesting to investigate. One of

those is the utility function that was defined in Section 2.2.2. It was already indicated that the utility function is not entirely accurate with respect to the preferences of microgrids: the demand and supply of prosumers could be represented by a function over time instead of a constant, the safety of errors could be modelled explicitly than just by size alone. Its formulation definitely has some room for improvement and could lead to more user-friendly microgrids. We hypothesize that as long as the new utility function does not result in larger microgrids, its usage will not affect the performance of our algorithm.

Another subject of interest would be the usage of pricing schemes to improve the social welfare. In this work, coalition stability was pursued using the quality of an obtained microgrid, here the cost of electricity was disregarded for the obtained utility. However by manipulating this cost, utility can perhaps be changed in such a way that unstable coalitions become stable. By properly manipulating the prices, social welfare could be improved without violating the constraint of stability.

Another thing we have not covered is a dynamic situation in the power grid. Demand and supply is constantly changing and even the network topology can change (connections can disappear, others appear). This would mean that a well formed microgrid could deteriorate over time. It would require constant awareness of the situation and the ability to adapt changes.

Perhaps a decent approach would be to maintain a hierarchy of coalitions instead of a single one (like it is now). Since every coalition is a combination of smaller coalitions, each individual coalition could maintain whether the merge of the sub-coalitions is still feasible and whether it should stay in its super-coalition or change. This would probably also make it easier to provide some means of backtracking in the formation of coalitions and further improve the social welfare.

On a final note, it might be possible to reduce the search area of the algorithm even further. While k-stability already decreases the search area from an entire network to a small neighbourhood, the algorithm still evaluates every possible combination within that neighbourhood. Perhaps one can shrink it even further by excluding unpromising combinations either by deduction or the use of a heuristic. This might be the decisive step needed to make the algorithm truly scalable.

BIBLIOGRAPHY

[ASC] ASCI. *DAS: Distributed ASCI Supercomputer*.

[Bal+10] Stefan Balev et al. *GraphStream*. 2010.

[Bal04] Coralio Ballester. "NP-completeness in hedonic games". In: *Games and Economic Behavior* 49 (2004), pp. 1–30. ISSN: 08998256. DOI: 10.1016/j.geb.2003.10.003.

[BCR14] Filippo Bistaffa, J. Cerquides, and Juan Rodriguez-aguilar. "Anytime Coalition Structure Generation on Synergy Graphs". In: *International Conference on Autonomous Agents and Multiagent Systems* (2014), pp. 13–20.

[BJ02] Anna Bogomolnaia and Matthew O. Jackson. "The Stability of Hedonic Coalition Structures". In: *Games and Economic Behavior* 38.2 (2002), pp. 201–230. ISSN: 08998256. DOI: 10.1006/game.2001.0877.

[Bol96] M.H.J. Bollen. "Voltage sags: effects, mitigation and prediction". In: *Power Engineering Journal* 10.3 (1996), p. 129. ISSN: 09503366. DOI: 10.1049/pe:19960304.

[Bon00] A Bondi. "Characteristics of Scalability and Their Impact on Performance". In: *Proceedings of the 2nd international workshop on Software and performance* (2000), pp. 195–203. ISSN: 158113195X. DOI: 10.1145/350391.350432.

[BZD14] Michael Burr, Michael Zimmer, and Peter Douglass. *About Microgrids*. 2014.

[Chr99] Richard D. Christie. *Power Systems Test Archive*. 1999.

[Cla14] Megan Clark. *Aging US Power Grid Blacks Out More Than Any Other Developed Nation*. 2014.

[Die05] Reinhard Diestel. "Graph Theory. 2005". In: *Grad. Texts in Math* 101 (2005).

[Fan+12] Xi Fang et al. "Smart grid - The new and improved power grid: A survey". In: *Communications Surveys & Tutorials* (2012).

[Far10] H Farhangi. "The path of the smart grid". In: *Power and Energy Magazine, IEEE* february (2010).

[Git] Paolo Gittoi. *Coalition Formation Theory | coalitiontheory.net*.

[GKP89] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: a foundation for computer science*. Addison-Wesley, 1989, p. 244.

[HVW14] Martin Hoefer, D Váz, and Lisa Wagner. “Hedonic Coalition Formation in Networks”. In: *Twenty-Ninth AAAI Conference on Artificial Intelligence* (2014).

[LA02] R Lasseter and A Akhil. “The CERTS microgrid concept”. In: *White paper for Transmission Reliability Program, Office of Power Technologies, US Department of Energy* (2002), p. 29.

[Min+10] Gouki Mine et al. “Construction Method of Dynamic Microgrid by Using Optimized Grouping Method”. In: *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on* (2010).

[Mye77] Roger B Myerson. “Graphs and Cooperation in Games.” In: *Mathematics of Operations Research* 2.3 (1977), pp. 225–229. ISSN: 0364-765X. DOI: 10.1287/moor.2.3.225.

[Net12] NetBeheerNL. *Storingen*. 2012.

[ØN08] Jacob Østergaard and Je Nielsen. “The Bornholm Power System An Overview”. In: *Centre for Electric Technology, Technical University of Denmark* April (2008).

[PA12] G.A. Pagani and M. Aiello. *The Power Grid as a Complex Network: a Survey*. 2012. arXiv: 1105.3338.

[RJ08] Talal Rahwan and N R Jennings. “An improved dynamic programming algorithm for coalition structure generation”. In: *Proc 7th Int Conf on Autonomous Agents and Multi-Agent Systems* (2008), pp. 1417–1420.

[RPH98] Michael H. Rothkopf, Aleksandar Pekec, and Ronald M. Harstad. “Computitional Manageable Combinatitional Auction”. In: *Management science* 44 (1998), pp. 1131–1147.

[San+99] Tuomas Sandholm et al. “Coalition structure generation with worst case guarantees”. In: *Artificial Intelligence* 111 (1999), pp. 209–238. ISSN: 00043702. DOI: 10.1016/S0004-3702(99)00036-3. arXiv: 9810005 [cs].

[Sch08] Roderick Schwass. “Case studies of onsite energy systems for healthcare facilities”. In: (2008).

[Sho05] Mike Shor. *Game Theory Dictionary*. 2005.

[Slo16] N. J. A. Sloane. *Sequence A000110 in The On-Line Encyclopedia of Integer Sequences*. 2016.

[Tan11] Nobuo Tanaka. “Technology Roadmap Smart Grids”. In: *International Energy Agency* (2011), p. 52. DOI: 10.1007/SpringerReference_7300.

[The15] The Guardian. *Turkey power outage shuts down public transportation and half of provinces*. Istanbul, Mar. 2015.

[TW04] Kagan Tumer and David Wolpert. “A survey of collectives”. In: *Collectives and the design of complex systems* (2004).

[Vin+12] Meritxell Vinyals et al. “Coalitional energy purchasing in the smart grid”. In: *2012 IEEE International Energy Conference and Exhibition, ENERGYCON 2012*. 2012, pp. 848–853. ISBN: 9781467314541. DOI: 10.1109/EnergyCon.2012.6348270.

[VPJ12] Thomas Voice, Maria Polukarov, and Nicholas R Jennings. “Coalition Structure Generation over Graphs”. In: *Journal of Artificial Intelligence Research* 45 (2012), pp. 165–196. DOI: 10.1145/0000000.0000000. arXiv: arXiv:1410.6516v1.

[VRJ12] Thomas Voice, Sarvapali D Ramchurn, and Nicholas R Jennings. “On Coalition Formation with Sparse Synergies”. In: *Aamas 2012* (2012), pp. 223–230.

[Wil+16] Willis et al. *Aging Power Delivery Infrastructures*. CRC Press, 2016.

[Woe13] Gerhard J. Woeginger. “A hardness result for core stability in additive hedonic games”. In: *Mathematical Social Sciences* 65.2 (2013), pp. 101–104. ISSN: 01654896. DOI: 10.1016/j.mathsocsci.2012.10.001.