Mechanisms for Network Creation

Mechanismen zur Generierung von Netzwerken

Master's Thesis

submitted in partial fulfillment for the degree of

Master of Science

in IT-Systems Engineering

Submitted by: **Stefan Neubert**

Supervisor: Prof. Dr. Tobias Friedrich
Advisor: Dr. Pascal Lenzner

of the Chair for Algorithm Engineering

Potsdam, 30.05.2018

Abstract

We introduce and analyze the possibilities of a new model for network creation by autonomous selfish agents: Unlike in typical network formation games such as the well-known model of Fabrikant et al. [25], the final network is not directly shaped by the players of a game. Instead, we design mechanisms that take edge preferences of agents as input for a social choice function and return a network that respects those preferences. In addition, it caters for compliance with global restrictions on the network topology and tries to establish several properties, such as global efficiency, maximizing the individual utility of agents, and building stable networks, especially in comparison to the result of an anarchic network formation. The mechanism also aims to produce Nash equilibria and encourages agents to honestly declare their preferences instead of playing strategically.

Specifically, we suggest three social choice functions for unweighted and one for weighted tree creation. We analyze them with regard to the introduced properties in the context of four different utility functions. Highlighting the power of the new model, we conclude with a mechanism for bounded-degree networks that performs considerably better than an uncoordinated network formation.

The mechanism approach is a true superset of both centralized network design and uncoordinated network creation games. To the best of our knowledge this is the first attempt to explore the realm inbetween those extremes. As our work is highly explorative, we do not yet attempt to find a perfect mechanism for all properties and network types, but lay out several suggestions for future work on this new model for network creation.

Zusammenfassung

Wir führen ein neues Modell zur Generierung von Netzwerken durch autonome, egoistische Spieler ein, und analysieren dieses: Im Gegensatz zu typischen Spielen zur Netzwerkerstellung, wie dem bekannten Modell von Fabrikant et al. [25], gestalten die Spieler das Ergebnisnetzwerk nicht direkt. Stattdessen entwickeln wir Mechanismen, welche die Kantenpräferenzen von Agenten als Eingabe für eine soziale Entscheidungsfunktion entgegennehmen, und ein Netzwerk zurückgeben, das diese Präferenzen berücksichtigt. Zusätzlich sorgt der Mechanismus für die Einhaltung globaler Einschränkungen an die Netzwerkstruktur und versucht, nützliche Eigenschaften zu erfüllen; zu diesen zählen die globale Effizienz des Netzwerks, die Maximierung des individuellen Nutzens aller Spieler, sowie die Konstruktion stabiler Netzwerke, vor allem im Vergleich zum Ergebnis eines unkoordinierten Prozesses zur Netzwerkgenerierung. Auch zielt der Mechanismus darauf ab, Nash-Gleichgewichte zu erzeugen, und fördert, dass Agenten ihre Präferenzen wahrheitsgetreu mitteilen, anstatt strategisch zu spielen.

Konkret schlagen wir drei soziale Entscheidungsfunktionen für ungewichtete, und eine für gewichtete Bäume vor. Wir analysieren diese dann im Bezug auf die eingeführten Eigenschaften im Kontext von vier verschiedenen Nutzenfunktionen. Abschließend stellen wir den Vorteil des Ansatzes mit Mechanismen heraus, indem wir für Netzwerke mit Knotengradbeschränkung einen Mechanismus präsentieren, der wesentlich bessere Ergebnisse liefert, als ein unkoordinierter Prozess zur Netzwerkerstellung.

Der neue Ansatz, Netzwerke mit Mechanismen zu erstellen, ist eine echte Obermenge bekannter Verfahren wie der zentralen Konstruktion von Netzwerken und den unkoordinierten, dezentralen Spielen zur Netzwerkgenerierung. Soweit uns bekannt, ist dies der erste Versuch, den Bereich zwischen diesen zwei Extremen zu erforschen. Die Arbeit ist darauf ausgerichtet, initial die Möglichkeiten des neuen Modells zu untersuchen. Daher versuchen wir noch nicht, einen perfekten Mechanismus zu präsentieren, der alle Eigenschaften und Netzwerktypen unterstützt. Stattdessen schlagen wir verschiedene Ansatzpunkte für zukünftige Forschung an diesem neuen Modell zur Netzwerkerstellung vor.

Contents

1. Introduction and Related Work

Networks can be seen as the universal structure of our age. While networks for infrastructure, for example for transportation, electricity, water and sewage, have always been of major importance, virtual networks without a primary physical representation now dominate our lives: Most importantly, the Internet as network of communication networks, but also all other collaboration and social interaction, processes and hierarchies, market relationships and resource management can be thought of and analyzed as graphs. The corresponding graph theory provides us with numerous insights into finding properties of or operating on such networks.

Some of those networks have been planned and built deliberately to form a specific topology. Others have grown naturally and still seem to incorporate several reoccurring properties — but we have so far not been able to fully understand how these properties come to be. As algorithm engineering can make use of those properties to develop faster solutions to everyday problems, building models that mimic real-world networks is a crucial task.

Implementations of such models could in addition help to build better networks for future requirements. This thesis explores a new approach by applying mechanism design techniques to the problem of network creation. We will start by summarizing existing algorithms for creating networks.

1.1. Central Authority

Some types of networks, such as Local Area Networks, water pipes in a new building or communication protocols for emergency services, are designed by a single central authority, such as an IT administrator, the building's architect or a legislative organ.

The designer of the network decides on the topology based on a global metric, for example she might optimize the number of edges, distances or robustness against edge or node failure. Usually, such networks are connected, thus all participants can interact with all others and no entity is isolated.

1.1.1. Standard Network topologies

One of the typical topologies of such networks is a *star* (Figure 1.1, a), in which one entity in the graph is charged with organizing all network traffic of all other participants. This structure provides short paths of length two between all pairs of nodes. Routing in such a network is easy, as the central node knows all its neighbors and can directly relate traffic to the respective recipient. However, this router bears all traffic load, and if it fails, no connectivity whatsoever remains.

If the designer adds additional layers to the star, a *hierarchy* forms (Figure 1.1, b), in which there are several "more important" nodes that route traffic between subtrees of the network. As this is more efficient than a single router, a hierarchical network can grow larger. Should one of those nodes fail, the graph breaks apart in several components. Nodes in those components are then no longer able to interact with the rest of the network, but might still be connected to some other participants.

In contrast to the first two structures, a *ring* of equal nodes (Figure 1.1, c) is robust against failure of one node or edge. It is in fact the sparsest 2-connected network, meaning that no network with fewer edges remains connected if a node or an edge fails. However, some distances between pairs of nodes are linear in the number of nodes in the network.

Being fully connected, *cliques* (Figure 1.1, d) provide the highest possible robustness and the shortest possible interaction path lengths. Though, the high node degrees and high number of edges can in some settings come with enormous costs and efforts for the participating nodes.

Except for the trees, those network topologies assume, that (apart from routers) participants in a network are of equal importance and are interested in the overall service quality but do not care about their exact placement in the graph. In the case of trees, the designer can implement a structural grouping that respects for example spatial conditions or some information on the interaction frequency distribution.

1.1.2. Algorithmic Network Design

Especially if the network in design has a spatial representation (in contrast to being purely virtual), a design algorithm can be used to optimize some measurement of quality. Such a design process can be thought of as a selection of one network from the set of all possible graphs with the given nodes.

If for example the target network is to be a tree and all possible edges between nodes are weighted, Prim's algorithm [50] or Kruskal's algorithm [38] can

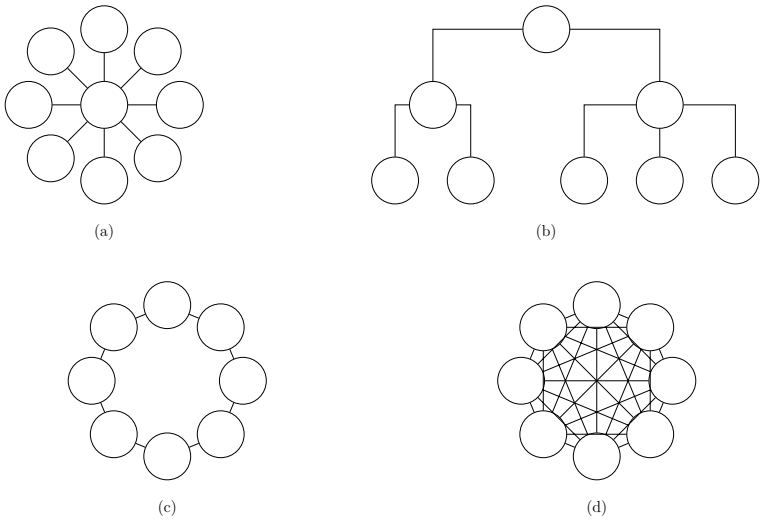


Figure 1.1.: Stars (a), (tree-)hierarchies (b), rings (c) and cliques (d) are standard network topologies, especially in IT networks.

be used to efficiently obtain a Minimum Spanning Tree, a tree that connects all nodes and minimizes the sum of edge weights.

Even small changes to optimization criteria tend to lead to intractable algorithmic problems. As an example, consider the following Problem: Given a set of nodes and an edge weight function, is there a subset of edges, whose total weight does not exceed a specified budget, that produces a k-connected graph? For $k = 1$ the problem can be solved by calculating the Minimum Spanning Tree. But for any fixed $k > 1$ the problem is NP-complete [27, Appendix 1.2, GT31]. For $k = 2$ this can easily be proven, as with all edge weights set to 1 the problem is equivalent to finding a Hamilton Circuit.

Similarly, consider the (simple) network design problem as stated by Johnson et al. [36] that again takes a set of nodes and an edge weight function as input. This time, the algorithm is tasked to decide whether there exists a graph in which the sum of the weights of the shortest paths between all node pairs is smaller than some threshold, and the sum of all edge weights does not exceed a specified budget. This problem is NP-complete even if all edge weights are one and the solution graph must be a tree. Garey and Johnson [27, Appendix A2] list several more NP-complete network design problems.

Due to the NP-completeness of many problems, approximate algorithms have been developed that trade time for precision, for example heuristics for networks that minimize the sum of shortest paths under the constraint of a overall budget [e.g. 22, 52].

Limitations of Centrally Designed Networks

For selected cases, a central authority might be able to design a network with good quality according to some metric. This metric usually is global and thus determines, whether the constructed graph is beneficial for the whole society. As with the Minimum Spanning Tree, it is sometimes even possible to achieve a social optimum, which is a state that yields the minimum/maximum value for the optimized metric. However, lots of network design problems become intractable for large numbers of nodes. Also, the bigger a network gets, the more unrealistic it seems, that a central authority could enforce the structure of a designed network.

1.2. Individual Actors

In fact, many real life instances such as the Internet, trans-regional transportation infrastructure or social networks are clearly not designed and implemented by any single authority, but are shaped by many uncoordinated actors. These actors no longer have global quality as a primary objective, but optimize their individual metric. Even more so: If modeled as a game, thus as an interaction between rational and selfish actors (called agents), the individual utility of an agent (also called cost if the utility is negative) is the only thing an agent optimizes by making decisions (called strategy) in that game.

The output network is then determined by the strategies of all agents. Either those decisions are all declared at the same time without the possibility of reconsidering — a so called one-shot game — or an iterative process enables agents to dynamically adjust the network to their benefit.

As we no longer assume the presence of a powerful authority that enforces a network structure, agents can be tempted to modify the topology in order to improve their outcome. If no single agent can increase her utility by unilaterally changing her strategy, a network is considered stable and the corresponding strategies form a so-called *Nash equilibrium*. Otherwise, the network is unstable, as there is at least one agent with an incentive to deviate from her strategy. Note, that it is not guaranteed, that there always is a Nash equilibrium.[1]

1.2.1. The Network Creation Game

Fabrikant et al. [25] proposed a network creation game in which each node of

1 We are only considering pure-strategies, thus agents have to decide on a single strategy and cannot assign probabilities.

the network represents an agent who decides on a subset of other players to build an undirected edge to. The union of those edges forms the output graph. Each agent has to pay a constant α per built edge and bears the sum of the shortest path distances to all other players as additional cost. By choosing her strategy (thus, the subset of other players) well, each agent tries to minimize her costs.

Numerous variations of this and other network creation games have been studied, especially in regards to how (worst-case) equilibrium networks look like and how they perform in comparison to optimal networks [1, 2, 4, 20, 21, 31, 42, 44]. Amongst others are different models for edge creation and cost sharing [5, 15, 16, 17, 18, 34], utility functions [8, 11, 29] and studies of game dynamics [3, 41].

As it is unrealistic that every player is equally interested in interacting with each other player, Halevi and Mansour [32] have suggested and analyzed an extension to the network creation game of Fabrikant et al. [25]. In this extension, building an edge again costs α, but agents only suffer a distant cost for a subset of players, called their *friends*. Cord-Landwehr et al. [19] introduced a similar extension to the basic network creation game of Alon et al. [3]. Similarly to these extensions we will also claim, that agents do not value all other players as equally important partners. Section 3.4.1 will give additional reasoning to this assumption. In contrast to Halevi and Mansour [32] and Cord-Landwehr et al. [19], our model however won't assume that friendship is symmetric.

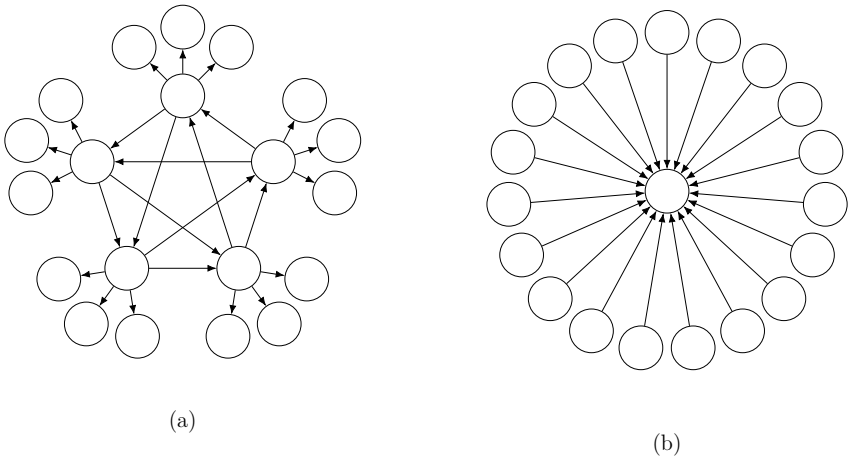
Like [7, 24, 39, 40], we will argue, that real-life actors might be subject to bounds on their node degree or budget. Lastly, as a lot of equilibrium networks turn out to be trees, concepts of network failure have been introduced to incentivize agents to build sparse but more resilient networks [12, 14, 30].

1.2.2. Inefficiency of Games

Even though network creation games closely reflect how many real life networks are created — most prominently the Internet graph — the results might suffer under the selfishness of agents. For example, a network game would yield a disconnected network, if global connectivity is not part of the utility function of agents. Moreover, not just the overall quality, fairness, but even the very utility an agent tries to maximize can be worse in an uncoordinated setting than in a controlled one. General game theory provides many examples for that, prominent ones being the Prisoners' Dilemma, the Hawk-Dove Game or the Tragedy of the Commons which are presented in Appendix A. Closer to our topic is a simple network routing game, that shows a similar effect as in the Prisoners' Dilemma and results in an unintuitive worsening of the agents'

utility due to their own selfish actions. This effect is known as Braess's paradox [13] and also presented in Appendix A.

In the network creation game of Fabrikant et al. [25], stable networks can also perform worse than designed networks — especially if one inspects the social cost, which here simply is the sum of the costs of all agents. Consider 20 agents, who selfishly create a network, in which they have to pay $\alpha = 4$ for each built edge, and suffer the shortest distance to all other players as additional cost. Should they form a (α, k) clique of stars as depicted in network (a) in Figure 1.2, it is stable for any clique size k, as shown by Albers et al. [2, Lemma 4.1]. However, the star shown in network (b) in the same figure performs better in terms of social cost and also individual cost for the clique nodes of (a). More specifically, the social cost of the clique of stars is 990, whereas the star only accounts for 798 in edge and distance cost.



(a) (b)

Figure 1.2.: Both networks are stable for $\alpha = 4$, but both social and some individual costs are higher for the $(5, 4)$ clique of stars graph (a) than for the star (b).

A widely used measurement for this loss in quality is the *Price of Anarchy* introduced by Koutsoupias and Papadimitriou [37]. This ratio compares the worst possible Nash equilibrium[2] to the social optimum and thus gives an idea of how much worse stable results can possibly be due to the lack of coordination. Both instances are either represented by a social cost or a social welfare value.

Fabrikant et al. [25] show, that for their model the price of anarchy is $\mathcal{O}(\sqrt{\alpha})$ for $\alpha < n^2$, and at least $3 - \epsilon$ for any $\epsilon > 0$ and large enough n. Exact values for the Price of Anarchy in the model are not yet known for all values of α. For

2 For games in which there might not exist stable results, Berger et al. [9] propose a measure of inefficiency that instead relies on expected values based on a defined or random order of agents that are allowed to make changes to their strategy.

large ranges it is known to be constant, as all equilibria for those α are trees [44]; for even more values this is conjectured. Bilò and Lenzner [10] presented a recent survey of the progress on this open *tree conjecture* and improved the best known bound to $\alpha > 4n - 13$.

In order to provide a valuable insight, one has to choose wisely how to assign a social cost or social welfare to a result. Obvious candidates include the sum of all agents' utilities or the minimum utility of an agent in the game, but game-specific values are also possible.

1.3. Mechanism Design

Between the centralized perspective of a governing authority or a private owner, and the distributed anarchic process of an uncoordinated game, mechanism design can be seen as a middle way: While the overall process still considers selfish, rational players and their strategies, the outcome is no longer determined directly by those strategies; instead agents have to stick to a set of game rules. One can try to design those rules in such way, that even though the agents' choices might still primarily define the outcome, a central authority can be able to impose additional requirements on the result of the game. As Nisan et al. [46, chap. 9.3.1] state, these rules can be seen as a concrete implementation of an *invisible hand* that guides players towards socially good behavior.

1.3.1. The Structure of Mechanisms

More formally, in the context of mechanism design, each player has a private preference about the output, based on which she publicly declares her strategy. A social choice function then takes all strategies as input and produces the outcome. As the exact function is known to all players, they can choose their strategy in such way that it optimizes their personal utility which is based on the outcome itself and an possible payment or fee by the mechanism.

When designing a mechanism one has to define allowed inputs for the players, possible outputs of the social choice function, and the function itself. As agents often can freely choose to participate in a mechanism or not, it has to ensure that no agent can be worse off by taking part in, than by ignoring the mechanism, as we expect *individual rationality* from the agents. In order to provide benefit over an uncoordinated game, the algorithm additionally should guarantee some individual or social quality of the output. Usually a mechanism should encourage agents to play *truthfully*, thus to state their

true preference instead of declaring a tactical strategy to improve their utility. Lastly, a mechanism should not rely on external subsidies, but has to be able to finance payments to some players through fees from others. This property is usually called *budget balance*.

1.3.2. Applications of Mechanism Design

The development of mechanisms has especially been popular for auctions, elections and voting systems. Even though mechanism design theory is generally attributed to Hurwicz [35][3], one of the major results in the realm of voting systems was published before by Arrow [6] and later extended by Gibbard [28] and Satterthwaite [51].

Essentially *Arrow's impossibility theorem* [6] states, that if agents are to select from at least three alternatives by ranking them, there is no decision mechanism that satisfies all of a set of seemingly reasonable fairness criteria: Firstly, if an irrelevant alternative is added or removed from the election, the outcome must not change (*Independence of irrelevant alternatives*). Secondly, the order of any two alternatives must neither be defined by the vote of a single *dictator*, nor the mechanism itself (*Nondicatorship* and *Citizens' sovereignty*). Finally, no alternative x can fall below an alternative y in the outcome, if the only changes to the individual rankings were to improve x's position (*Positive Association of social and individual values*).

Impossibility theorems like Arrow's theorem unveil the limits of mechanisms in respect to their fairness. Nonetheless, there are equally important results covering the positive potential of mechanisms. For example, *Vickrey's Second Price Auction* [54] is a strategy-proof mechanism, meaning that no bidder can improve her chance of winning the auction by bidding a false value. The generalized concept of *Vickrey-Clarke-Groves Mechanisms* can be used in many contexts. Such a mechanism is able to select an outcome that maximizes the social welfare, which is the sum of all players' valuation of the outcome, whilst ensuring that every agent has an incentive to play truthful [46, chap. 9.3.3]. In 2009, Elinor Ostrom was awarded the Nobel Memorial Prize in Economic Sciences for researching several real-world non-governmental solutions of the Tragedy of the Commons; those solutions basically being community-based mechanisms [48].

Closest to our application of mechanism design to network creation is the well-known stable marriage problem posed and solved by Gale and Shapley [26].

3 Leonid Hurwicz, Eric S. Maskin and Roger B. Myerson have been awarded the Nobel Memorial Prize in Economic Sciences "for having laid the foundations of mechanism design theory"[47]

Even though not usually formulated in that nomenclature, their algorithm essentially is a mechanism: A set of n men and n women declare their marriage preferences as input to a social choice function, which then matches them in couples. The mechanism makes sure that the matching is stable in the sense, that there is no pair of a man and a woman who prefers each other over their assigned partner. We will build on the same idea of preferred partners, but construct a network of equal agents instead of a bipartite matching.

1.3.3. Algorithmic Mechanism Design

Mostly, mechanism design theory only deals with the properties of a mechanism's output, not with the computational aspects of the social choice function. More recently, mechanism design theory and algorithm analysis have been combined: Nisan and Ronen [45] coined the term *Algorithmic Mechanism Design* and suggested both to use mechanism design to solve "algorithmic problems in a distributed setting"[45] and to analyze whether mechanisms are *poly-time* computable.

Most importantly for us, all computational parts of a mechanism, which are both decisions made by the agents and the social choice function, are now subject to algorithmic complexity (worst-case) analysis. Mechanisms that can be implemented to run in polynomial time are of particular interest, as longer runtimes are considered unrealistic for real-world applications of a mechanism.

1.4. Mechanisms for Network Creation

This thesis explores the potential of mechanisms if used for network creation. Just like in the network creation game by Fabrikant et al. [25], participating agents represent nodes in the network-to-be. A mechanism will take preferences on neighbors from each agent as input and generate an output graph that tries to fulfill those preferences.

As a mechanism's way to produce that output graph is not restricted in any way, mechanisms are a true superset of both one-shot network games and centrally designed networks: If each agent's preferences are stated as subset of all other agents, and the mechanism simply returns a graph that contains exactly the edges that correspond to those preferences, the process is essentially the same as in the network game. By ignoring the input completely and generating an arbitrary network structure based only on the number of agents or some public information, the mechanism resembles a central authority that constructs the graph.

Our goal is to find mechanisms in the realm inbetween those extremes, that are able to build properties into the network desirable for individual agents or the society as a whole, and that are not achieved by either described "pure" form of network generation. We will focus on six properties a mechanism should fulfill. The reasoning for those properties follows here; we define them more precisely in Section 2.4.

Profitable Participation To effectively implement a mechanism, agents have to be willing to declare their neighbor preferences to the mechanism and accept the result. As no rational agent would risk to end up with a lower utility by joining such a network, the mechanism has to guarantee that each agent receives at least as much (or even more) utility as she would in a anarchic network creation process. For that, we will compare the individual utility of each agent in worst-case instances of mechanism-generated networks and worst-case results of the network game.

High individual Profit In addition to the lower bound on allowed utilities, the mechanism should not only limit risks, but even promise high reward for each agent. Preferably, agents achieve the utility of the (individual) best-case network. As it might not be possible for all agents to simultaneously get the maximum profit, the mechanism should at least guarantee to build a network that yields close-to-optimal utility for many agents.

Stable networks As long as an agent could increase her utility by deviating from the generated network, for example through dropping planned edges and buying different ones instead, it seems unlikely that all agents are happy with the mechanism's output. On the other hand, we call a network *stable*, if for some specified allowed deviations no agent can improve her profit. If such a stable network exists, a mechanism should choose it over unstable ones. This might not always be possible; in such cases the mechanism should aim for a network that is approximately stable, meaning that no agent can improve her utility by much.

Rewarding Honesty No agent should be able to manipulate the output network and increase her utility by lying about her true preferences. As a mechanism only knows the declared strategies and not the preferences, it cannot enforce honesty. Instead, it must incentivize honest strategies in such way, that it is always better (or at least as good) for an agent to tell the truth than to play tactically.

High social Welfare As the tragedy of the commons and the prisoners' dilemma show, dominating strategies might result in a worst-case outcome for both individual agents and the society. One of the main advantages of a mechanism over a game is, that it can try to avoid such an outcome by generating an output that optimizes the individual and social welfare — even if this output does not correspond to dominating strategies. At the same time, a mechanism can try to establish some kind of fairness among participating agents. We will show examples later, in which some have to endure low utility in order to maximize the utility of others. Rather, costs should be distributed among participants in a way that no agent can free-ride the network at the expense of others.

Global Properties A mechanism can also try to implement a structure that benefits the society, even though this benefit might not influence the utilities of individual agents. Such global properties are for example the guaranty, that a generated network is connected, even when the agents' preferences do not require it. Similarly, a small diameter or a robustness against edge- or node-failure is desirable.

Some of those properties might be achieved by simply comparing all possible (usually exponentially many) output graphs and selecting the best one. We require however, that mechanisms are feasible, thus they have to run in polynomial time.

This thesis explores several mechanisms and utility functions in order to show that for each of the six properties there are mechanisms which achieve them (see Table 1.1). It is not content of this work to find a single mechanism that achieves all of them, this is left as future work.

Social Choice function	Utility function	Individual Rationality	Efficiency global/individual	Stability swap/anarchy	Truthfulness
Iterative Attachment	Dist. Costs	no	no/no	no/no	yes
	Bounded Dist.	yes	no/no	no/no	yes
	Profit from all	trivially yes	—	no/—	—
	Exp. Decrease	trivially yes	—	no/—	—
Cycle Breaking	Dist. Costs	for most	yes/for most	rarely/yes	yes
	Bounded Dist.	yes	no/for most	sometimes/yes	yes
	Profit from all	trivially yes	—	rarely/—	—
	Exp. Decrease	trivially yes	—	rarely/—	—
Cycle Replacement	Dist. Costs	mostly for all	yes/2-approx for all	rarely/better	yes
	Bounded Dist.	yes	yes/yes	always/better	yes
	Profit from all	trivially yes	—	sometimes/—	—
	Exp. Decrease	trivially yes	—	rarely/—	—
Greedy MST	Ranked	—	some/—	—	no
Snowflakes	—	—	—	—/much better	—

Table 1.1.: Overview of all social choice functions, utility functions and properties. The entries are heavily simplified; see the respective chapters for exact statements. Dashes mark properties not analyzed in this work.

2. Definitions and Notation

After informally introducing the concepts of mechanisms and network creation, we will now define our model and related properties.

2.1. Mechanisms

All mechanisms in this thesis consider a set of n agents $A = \{a_1, \ldots, a_n\}$. Each agent a_i has private preferences that we call the type $t_i \in T_i$ of the agent, where T_i is the set of possible types of agent a_i. We denote the vector of all types by $t = (t_1, \ldots, t_n)$ and the vector of the types of all agents except for a_i by $t_{-i} = (t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n)$. As a shorthand we will write (t_i, t_{-i}) for t.

While the type of each agent a_i is fixed, she freely chooses her strategy s_i from all of her possible strategies S_i and publicly declares it. Usually, we will chose $S_i = T_i$ for simplicity. Analogous to the agents' types we denote the vectors s, s_{-i} and (s_i, s_{-i}).

Taking the strategy vector s as input, the mechanism's social choice function f will return some outcome $o \in O$, where O is the set of all possible outcomes. If the mechanism decides to pay some compensation to or charge a fee from agents, f will additionally return a sequence of payments $(p_i)_{i=1}^n$. A positive value p_i describes a payment from the mechanism to the agent, a negative value is a payment from the agent to the mechanism.

Each agent a_i then derives some value $v_i(o) \in \mathbb{R}$ from the outcome. By choosing her strategy, she tries to optimize her utility $u_i(o) = v_i(o) + p_i$. This implies that we only consider *quasilinear preferences*, meaning both the value and the payment can be measured in money.

All combined, a mechanism M consists of a sequence of strategy sets $(S_i)_{i=1}^n$, a set of possible outcomes O and a social choice function $f : (S_1, \ldots, S_n) \to O$ or (with payments) $f : (S_1, \ldots, S_n) \to O \times \mathbb{R}^n$.

2.2. Mechanisms for Network Creation

In our context, based on the strategy vector s, mechanisms will produce an output graph $o(s) = G = (V, E)$, in which the participating agents A are

represented by vertices $V = A$. We will therefore use the terms *agent*, *vertex* and *node* interchangeable. Remember, that a mechanism is not entirely free to construct $o(s)$, but has to produce some $o(s) \in O$.

A strategy of an agent always describes a set of desired edges from the agent to her preferred neighbors. The whole strategy vector s therefore resembles a *strategy graph* G_s. We generally distinguish two categories:

In the unweighted case (see Figure 2.1), a strategy s_i is a subset of all other agents $A \setminus \{a_i\}$. The set s_i is then called the set of a_i's *friends*. Consequently, the edges of the strategy graph $G_s = (A, E)$ are given by $E = \{(a_i, a_p) \mid a_i \in A, a_p \in s_i\}$. If $|s_i| = 1$, we will usually omit the set braces.

Alternatively, agents can be allowed to rank other agents in order of decreasing importance to them (see Figure 2.2). Strategies s_i are then given as tuple of k distinct preferred neighbors. The strategy graph $G_s = (A, E, w)$ is augmented with an edge-weight function $w : E \to \{1, \ldots, k\}$. If r is the rank of agent a_j in the strategy tuple s_i of agent a_i, then $w(a_i, a_j) = r$. We denote this agent a_j with rank r in the strategy of a_i by $s_i[r]$, analogously we use $t_i[r]$. The edge set E is defined the same as above.

Agent	Strategy
a_1	$\{a_2, a_3\}$
a_2	$\{a_3, a_4, a_5\}$
a_3	$\{a_4, a_2\}$
a_4	a_5
a_5	$\{a_1, a_2\}$

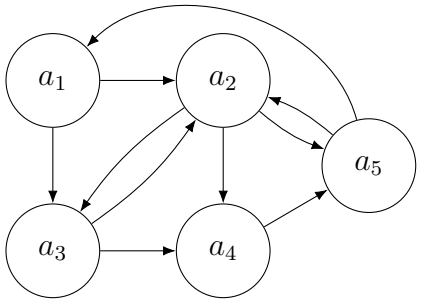


Figure 2.1.: An example for unweighted strategies s (left) and the corresponding strategy graph G_s (right).

The direction of an edge $e \in E$ shows, which agent of the two endpoints wished or even paid for the edge. An edge $e = (a_i, a_j)$ is considered to be directed from a_i to a_j. If edges are paid for by agents, e is owned by agent a_i, otherwise the direction simply shows, that a_i initiated the connection by including a_j in her strategy. However, in terms of connectivity we will ignore the direction; the edge can be equally used by both adjacent agents to derive value.

We will use set operations as notation for changes to a graph. Thus, $G \setminus \{e\}$ describes the removal of edge e from the graph G, and $G \cup \{e\}$ its addition; same holds for vertices.

Agent	Strategy
a_1	(a_3, a_2)
a_2	(a_3, a_4)
a_3	(a_2, a_4)
a_4	(a_5, a_2)
a_5	(a_1, a_2)

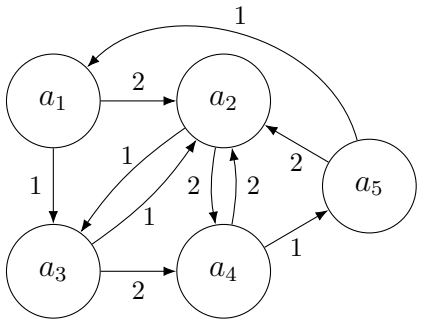


Figure 2.2.: An example for weighted strategies s with $k = 2$ (left) and the corresponding strategy graph G_s (right).

Finally, the undirected hop distance between two agents x and y in the graph G is denoted by $d_G(x, y)$; we define this distance to be infinite if x and y are not connected in G.

2.3. Anarchic Network Creation

To evaluate our mechanisms, we will compare their results with worst-case graphs created in an anarchic process. While the agents in this process have the same types and utility functions as in the mechanism, the network creation itself is uncoordinated. Just as with mechanism-created networks, the resulting graph however should be be a graph $o \in O$. Due to the lack of coordination, agents cannot be forced to build edges essential to the constraints of O, but we assume they can at least not add edges that would violate those constraints.

According to their utility function, the agents decide on a strategy that describes which edges they would like to establish. Our anarchic process starts with the empty graph $(A, \emptyset)$, to which the agents one-by-one add their preferred edges if they are conform with the conditions imposed by O. Each instance of that process is defined by the build-order of participating agents. A worst case instance thus corresponds to that permutation of agents, that produces the lowest utility for the society or the agent in question. As with the agents' strategies, we distinguish between unweighted and weighted edge preferences:

If an agent's strategy represents a set of friends of equal importance, she adds edges to all those friends at once, or at least to as many as possible without violating a graph constraint. This simulates, that an agent a_i wants all her friends as close as possible, and for that immediately tries to build direct edges.

In the ranked version, the agents play k rounds and try in each round to add an edge to the correspondingly ranked agent in their strategy tuple. Thereby, the process represents their preference of some agents over other possible neighbors in the network.

Both versions are similar to the network creation game proposed by Fabrikant et al. [25], in that the resulting graph is essentially the union of the agents' strategies. Only, in our case there are additional constraints on the result graph and the utility functions differ.

Note that neither of those processes guarantees to result in an equilibrium network. Even though it would be fair to compare mechanism outputs to equilibrium networks only, not all utility functions have an equilibrium in O, nor can it be guaranteed that an anarchic process reaches such an equilibrium state. A simple example sets O to the set of connected graphs; if the agents' strategies lead to disconnected components in the original network creation game, the two introduced processes would do so as well. Depending on the utility function, any additional or replaced edge that establishes connectivity might then lead to an unstable result.

2.4. Properties

As stated before, we will only consider mechanisms with feasible social choice functions:

Definition 1 (Poly-Time Mechanism)**.** A mechanism is *poly-time computable*, if and only if its social choice function can be implemented to run in polynomial time complexity.

We will always request, that a mechanism works without external subsidies, thus has to finance payments through fees:

Definition 2 (Budget Balance)**.** The outcome of a mechanism is *budget balanced*, if and only if the sum of all payments is non-positive, thus $\sum_{i=0}^{n} p_i \leq 0$.

If and only if a mechanism only produces budget balanced outcomes, it is called a *budget balanced mechanism*.

To ensure a good outcome for the society as a whole, mechanisms should try to maximize the agents' derived value:

Definition 3 (Efficiency and Social Welfare)**.** An outcome o is *efficient*, if and only if it maximizes the sum of values (called *social welfare*), thus for all other

outcomes $o' \in O \setminus \{o\}$ it holds that $\sum_{i=0}^{n} v_i(o) \geq \sum_{i=0}^{n} v_i(o')$.

A mechanism is considered *efficient*, if and only if it produces efficient outcomes for all possible strategy vectors s.

Maximizing the sum does not yet ensure that all agents actually derive value. As no agent can be expected to endure negative utility, mechanisms have to additionally ensure that agents participate voluntarily:

Definition 4 (Individual Rationality)**.** An outcome o is *individually rational*, if and only if no agent has a negative utility, thus $\forall a_i \in A : u_i(o) \geq 0$.

If and only if for all strategy vectors s a mechanism only produces individually rational networks, the mechanism itself is *individually rational*.

Additionally, the outcomes produced by mechanisms should at least be as good as the results of an anarchic network creation:

Definition 5 (Anarchy Stability)**.** The outcome o and the corresponding payments $(p_i)_{i=0}^{n}$ of a mechanism are *anarchy stable*, if and only if no agent has lower utility than their type would yield in the anarchic network creation defined in Section 2.3.

If and only if this holds for all strategy vectors s and the corresponding outcome o and payments $(p_i)_{i=0}^{n}$, we also call the mechanism *anarchy stable*.

Definitions 4 and 5 represent two different perspectives on the utility an agent derives by not participating in a mechanism. The first one assumes that an agent outside the mechanism has a utility of 0, as she neither earns something from interaction with other agents, nor does she have to maintain connections or pay for their usage. The second one considers that agents could be better off if they all together decide to play a network creation game without coordination. Effectively these stabilities assume that agents do not submit to a mechanism when they would be strictly better of without it. On the other hand, if they risked a decrease in utility by building a network themselves, they would prefer the mechanism.

An even stronger property is that of individual high profit:

Definition 6 (Individual Efficiency)**.** The outcome o of a mechanism is *individually efficient* for a set of agents $A^* \subseteq A$, if and only if it maximizes the individually derived value for those agents; thus for all other outcomes $o' \in O \setminus \{o\}$ it holds that $\forall a_i \in A^* : v_i(o) \geq v_i(o')$. If and only if there is a constant c such that $\forall a_i \in A^* : v_i(o) \cdot c \geq v_i(o')$, the mechanism is *c-approximately*

individually efficient for A^* and c.

A mechanism is *(approximately) individually efficient*, if and only if for all strategy vectors s its social choice function f produces an outcome $o = f(s)$ that fulfills this property.

A weaker property does not require that the result is a global optimum for the agents, but only requires a local optimum in respect of an agents ability to change her direct neighborhood:

Definition 7 (Swap Stability)**.** A resulting network o is *swap-stable*, if and only if no agent can strictly increase her utility by replacing an adjacent edge with an edge to some other player in the network. It is *c-approximatively swap-stable*, if and only if no agent can strictly increase her utility by more than a constant factor c.

Note that an agent must not violate structural limitations of the network by swapping some edge; thus the new resulting network o' after the swap must still be a valid result from the set of all possible outcomes O. This stability criterion already implies a detail of how connections can be established or destroyed in the network: If one of the two agents that form the endpoints of an edge should find it beneficial to cancel that edge, she can do so without having to ask the other agent for permission, in the same way as Alon et al. [3] define edge swaps in their basic network creation game.

Finally, mechanisms should encourage agents to play truthfully, thus to specify their true type as strategy:

Definition 8 (Truthfulness)**.** A mechanism is *truthful*, if and only if for any strategy vector (s_i, s_{-i}) the social choice function f produces an outcome o and payments $(p_i)_{i=0}^n$, such that no agent a_i can increase her utility by playing a strategy $s_i \neq t_i$, thus it holds that $\forall a_i \in A, \forall s_i \neq t_i : u(f(t_i, s_{-i})) \geq u(f(s_i, s_{-i}))$.

3. Simple Mechanisms for Trees

The first mechanisms we will design and analyze are restricted to trees, thus the set of possible outcomes O is the set of all trees with the agents A as nodes. For each agent a_i, the set of possible types T_i is defined by $T_i = A \setminus \{a_i\}$, meaning that each agent has exactly one friend she wants to interact with. As we choose $S_i = T_i$ for all agents, the strategy an agent declares also consists of one other agent she wants to be close to in the resulting network.

We consider three simple social choice functions, and four different utility functions to illustrate the properties introduced in Section 2.4. For the first utility function, we analyze all properties in detail, whereas we concentrate on interesting results for the later ones.

3.1. Three Social Choice Functions

All functions presented here make use of some random selection of agents. This selection should be made uniformly at random. However, as we mainly analyze worst-case scenarios, we will assume that the mechanism always selects the first agent from $(a_1, \ldots, a_n)$ that fulfills the respective property and construct our instances in such way, that this "random" selection leads to the intended worst-case result.

3.1.1. Iterative Attachment

Starting with an arbitrary agent, the function iteratively attaches the remaining agents to the resulting tree. If any of those remaining agents prefers to build an edge to some agent already in the tree, it is selected to extend the network. In the case that there is no extension of the tree possible based on the declared strategies, some arbitrary agent from the remaining ones is selected to build an edge to some arbitrary agent in the current tree. The function is presented in Algorithm 3.1.

The social choice function can be implemented to run in $\mathcal{O}(n)$ by maintaining a list of possible next agents to attach: Whenever an agent a_i is attached to the tree, each agent that declares a_i as her strategy is added to this list. This

Algorithm 3.1: Iterative tree construction

Input: Agents $A = \{a_1, \ldots, a_n\}$, Strategy vector $s = (s_1, \ldots, s_n)$
Output: Generated tree

$A^0 \leftarrow A$;
$G^0 \leftarrow \emptyset$;
for $r = 1$ *to* n **do**
  if $\exists a_i \in A^{r-1}$ *with* $s_i \in G^{r-1}$ **then**
    $G^r \leftarrow G^{r-1} \cup \{a_i\} \cup (a_i, s_i)$;
  else
    choose some $a_i \in A^{r-1}$;
    $G^r \leftarrow G^{r-1} \cup \{a_i\}$;
    if $G^{r-1} \neq \emptyset$ **then**
      choose some $x_i \in G^{r-1}$;
      $G^r \leftarrow G^r \cup (a_i, x_i)$;
  $A^r \leftarrow A^{r-1} \setminus \{a_i\}$;
return G^n

can be done in amortized time $\mathcal{O}(1)$ per agent by looking up those agents in an adjacency list of the transposed strategy graph G_s^T. In order to implement the introduced tie-breaking for random choices one can use a priority queue instead of a list and achieve a total complexity of $\mathcal{O}(n \log n)$.

Clearly, the mechanism therefore is *poly-time computable* when using Algorithm 3.1 as social choice function. As it does not issue any payments, it is also *budget balanced*.

3.1.2. Cycle Breaking

The second social choice function starts with the graph G_s that is formed by the agents' strategies. This graph most likely consists of several disjoint components. By pigeonhole principle there exists exactly one cycle in each of those components. To connect the components, some arbitrary agent inside such a cycle is selected and its edge removed from the graph. If the graph still consists of multiple components, the mechanism introduces a new edge from this agent to some arbitrary agent of some other component.

The function is presented in Algorithm 3.2 where $CC(x)$ denotes the connected component that contains agent x.

As the number of edges in G is in $\Theta(n)$, both finding connected components and cycles can be done in $\mathcal{O}(n)$. Thus, the function can be implemented to

Algorithm 3.2: Cycle breaking

Input: Agents $A = \{a_1, \ldots, a_n\}$, Strategy vector $s = (s_1, \ldots, s_n)$
Output: Generated tree

$G^0 \leftarrow G_s$;
$c \leftarrow$ number of components in G^0;
for $r = 1$ *to* c **do**
    choose some $a_i \in A$ that is part of a cycle;
    $G^r \leftarrow G^{r-1} \setminus (a_i, s_i)$;
    if $r < c$ **then**
        choose some $x_i \in A$ with $CC(x_i) \neq CC(a_i)$;
        $G^r \leftarrow G^r \cup (a_i, x_i)$;
return G^c

run in linear time. This holds even for the mentioned tie-breaking, as both for components and cycles the respective "arbitrary" agent can be determined in one go with finding those components and cycles.

Again, the mechanism is *poly-time computable* when using Algorithm 3.1 as social choice function; it is also *budget balanced*.

3.1.3. Cycle Replacement

The third social choice function is based on the same idea as the second one. Differently to the adjustments to G_s made there, this function however does not eliminate cycles by removing one edge, but transforms them into stars with one arbitrary agent of that cycle as center, as demonstrated in Figure 3.1.

The function is presented in Algorithm 3.3 where $CC(x)$ again denotes the connected component that contains agent x.

As every edge of each cycle in G_s is transformed exactly once, Algorithm 3.3 has the same complexity as Cycle breaking and therefore runs in linear time. Analogously, the mechanism is *poly-time computable* and *budget balanced*.

3.2. Constant Profit with Distance Costs

Whereas the first two properties — poly-time computability and budget balancedness — are independent of the value agents make, the remaining six properties depend on a defined utility function. As payments so far are always zero, we only need to look on the value functions v_i of the agents.

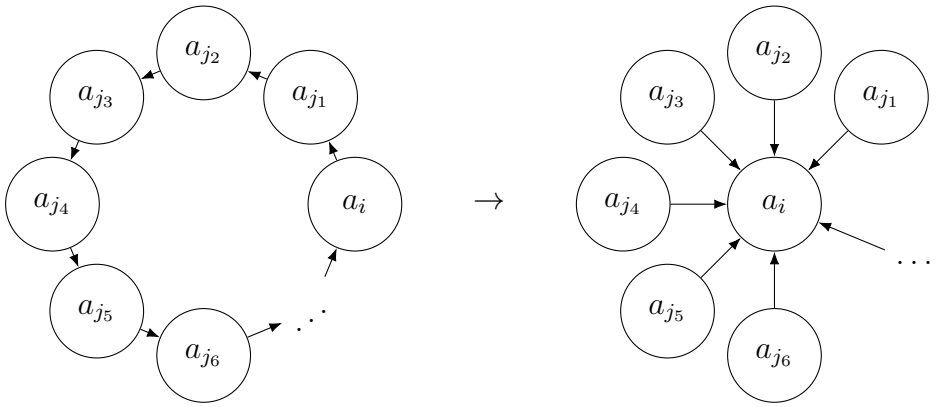


Figure 3.1.: The third social choice function replaces cycles with stars.

Algorithm 3.3: Cycle replacement

Input: Agents $A = \{a_1, \ldots, a_n\}$, Strategy vector $s = (s_1, \ldots, s_n)$
Output: Generated tree

$G^0 \leftarrow G_s$;
$c \leftarrow$ number of components in G^0;
for $r = 1$ *to* c **do**
  choose some $a_i \in A$ that is part of a cycle C;
  construct star S from agents in C with a_i as center node;
  $G^r \leftarrow (G^{r-1} \setminus C) \cup S$;
  if $r < c$ **then**
    choose some $x_i \in A$ with $CC(x_i) \neq CC(a_i)$;
    $G^r \leftarrow G^r \cup (a_i, x_i)$;
return G^c

We start by assuming that an agent a_i derives some constant value $\beta \in \mathbb{R}, \beta > 1$ from interacting (e.g. communicating) with her preferred partner t_i. However, she has to pay $d_o(a_i, t_i)$ for using the edges on the path to her partner. The first utility function we analyze thus is defined as

$$u_{i,1}(o) = v_{i,1}(o) = \beta - d_o(a_i, t_i). \tag{3.1}$$

Effectively, the function limits the maximum distance that an agent finds acceptable for interacting with her preferred partner. All agents therefore try to optimize the worst-case distance to their respective friend, as they only derive the maximum value of $\beta - 1$ by being directly adjacent to their respective friend.

For all but the analysis of truthfulness itself, we will assume that agents declare their true type as strategy, thus that $s = t$.

3.2.1. Social Welfare and Efficiency

As all three social choice functions always produce a connected tree, the social welfare is maximized if and only if the sum of distances between the players and their respective preferred partners is minimized. For a resulting tree o we define the social distance to be the sum of the undirected distances between all players a_i and their preferred partners t_i. Remember that $d_G(x, y)$ is the undirected hop distance between two agents x and y in the graph G, then we define

$$\text{Dist}(o) = \sum_{i=1}^{n} d_o(a_i, t_i).$$

Clearly $\text{Dist}(o) \geq n$, as the distance between some agent a_i and her preferred partner t_i is at least 1. (As Theorem 3.1 shows, the exact value for this social optimum is determined by the number of cycles and agents therein in G_s.) Similarly, $\text{Dist}(o) \in \mathcal{O}(n^2)$, as the maximum distance between two agents is $n - 1$.

Theorem 3.1. *The social optimum distance for a resulting tree o is $n + n_c - 2c$ where n_c is the number of agents that are part of cycles in G_s and c is the number of cycles in G_s.*

To prove this theorem, we will show first, that breaking a cycle into a path achieves the optimal social distance for the agents in that cycle.

Lemma 3.1. *For a set of agents A' that form a cycle C in G_s, the optimal summed distance in an output tree is $2 \cdot |C| - 2$, where $|C| = |A'|$ is the number of nodes in the cycle. This distance can be achieved by removing exactly one arbitrarily chosen edge e from the cycle.*

Proof. Note that due to symmetry, e can be chosen arbitrarily. Thus, if the lemma holds for some edge, it also holds for any edge.

For $|C| = 2$ the lemma follows trivially, as there is only one possible output tree and it fulfills the condition. Thus, assume $|C| > 2$.

If there is exactly one agent in an output tree, that is not adjacent to her preferred partner, the tree is a path in the described form and thus the lemma holds, as this agent has a communication cost of $|C| - 1$ and the other $|C| - 1$ agents only have to pay 1 to interact with their preferred partners. Note that there cannot be less agents not adjacent to their preferred partners. Thus

consider some arbitrary output tree that achieves the optimal social distance, in which there are at least two such agents.

Root this tree on some node r. Among the agents not adjacent to their preferred partners, select one agent a with maximum distance to r. Due to the selection of a, all agents in a's subtree are adjacent to their preferred partners. As the original graph was a cycle, this implies that the subtree underneath a is a path and there is exactly one agent a' outside that subtree whose preferred partner is in that path. (This preferred partner is the subtree's leaf.)

Let d be the distance between a and her preferred partner. Replace the edge between a and her parent with an edge to her preferred partner. By construction, for all agents except for a and a', the distance to their respective preferred partners does not change. However, the distance for a is reduced by $d-1$ whereas the distance of a' to her preferred partner is increased by at most $d-1$. Therefore, the social distance of the tree does not increase, and the new tree as well yields the optimal social distance.

Repeat this process until r is the only agent not adjacent to her preferred partner, and the final tree is a path in the described form. This must happen after at most $n-1$ steps, as the number of agents whose preferred partner is their direct parent strictly increases by one each time an edge is replaced in the described way. As in the final path there are $|C|-1$ agents with a distance of 1 to their respective preferred partner, and one agent with a distance of $|C|-1$, the optimal summed distance for a cycle is $2 \cdot |C| - 2$. □

With this, we now can derive the social optimum distance a mechanism can achieve for a strategy vector s and its induced Graph G_s.

Proof of Theorem 3.1. As the social distance only depends on the distance between agents and their preferred partners, each connected component of G_s can be dealt with independently. By pigeonhole principle, a connected component contains exactly one cycle, for which by Lemma 3.1 a path yields the optimal social distance. The remaining agents, that are not part of the cycle, can all be connected directly to their respective preferred partners and thereby produce the minimal possible distance of 1 per agent.

Thus, in the social optimum for an instance with n agents that has c cycles in G_s which are formed by (in total) n_c agents, there are $n-c$ agents with a distance of 1. In addition, in each cycle C there is one agent with a distance of $|C|-1$ to her preferred partner, which yields a total distance of $n+n_c-2c$. □

We say a social choice function achieves some worst case or best case social distance, if there exists at least one instance on which it produces an output o with such distance.

Iterative Attachment

Even though the first, overly simplistic mechanism can produce good results in some rare input cases, it does not generally guarantee anything better than the worst possible outcome. This is due to the fact that the iterative attachment can fail for a series of agents and thereby produce disadvantageous networks through bad random choices.

Theorem 3.2. *Iterative attachment achieves a best case social distance of n and a worst case of $\Theta(n^2)$. In the worst case, agents can have a distance of $\Theta(n)$ to their respective preferred partner.*

Proof. If G_s is a star with one additional edge (as there are n edges in G_s) and the center is picked first, the resulting tree o has $\mathrm{Dist}(o) = n$ as all agents are adjacent to their respective preferred partner.

However, if G_s consists of a star and an attached cycle like the input graph shown in Figure 3.2, the resulting tree o has $\mathrm{Dist}(o) \in \Theta(n^2)$, as the agents $a_1, \ldots, a_{\frac{n}{2}}$ have a distance of at least $\frac{n}{2}$ to their preferred partner a_n.

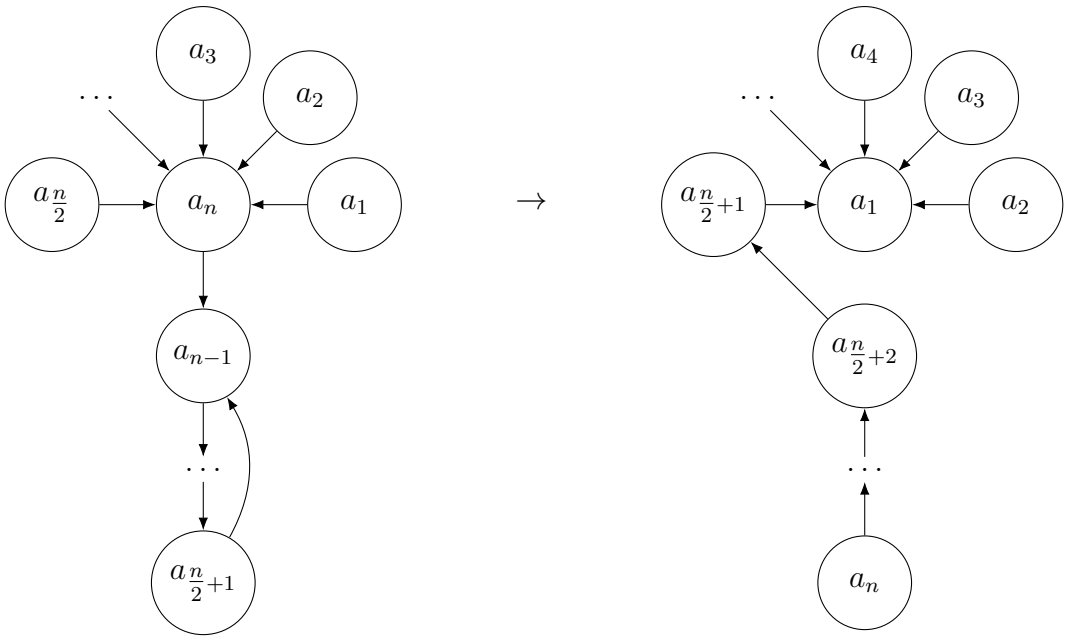


Figure 3.2.: A worst case instance for Iterative attachment with quadratic social distance.

□

Theorem 3.2 establishes that, even though some output networks can be both *efficient* and *individually efficient*, the mechanism as a whole is not. As

individual distances can be linear, and the overall distance sum quadratic, Iterative attachment does not even accomplish to always produce *(individually) approximately efficient* networks. While this is only a worst-case statement it still means, that agents must fear that a result network performs bad for both themselves and the whole society.

Cycle Breaking

In contrast to the first social choice function, the second one guarantees to construct networks with optimal social distance.

Theorem 3.3. *Cycle breaking achieves a best and worst case social distance of* $\mathrm{Dist}(o) = n + n_c - 2c$ *and thus achieves the social optimum distance. In the worst case, agents can have a distance of* $\Theta(n)$ *to their preferred partner, in the best case all are directly adjacent.*

Proof. Cycle breaking always exactly produces the graph described in Theorem 3.1. If the original graph G_s is a cycle with all n agents in it, removing one edge leaves all but one agent directly adjacent to their respective preferred partner, whilst one agents bears a distance of $n - 1$ to hers. For all strategy graphs G_s that do not include cycles with more than two agents, breaking these cycles leaves all agents adjacent to their respective friend. □

By Theorem 3.3, all output networks are *efficient*, and thereby the mechanism is as well. All networks are *individually efficient* for all but one agent per cycle with size greater two; for those they are however not even *approximately individually efficient*.

Cycle Replacement

The second social choice function therefore produces a network with optimal social distance, but individual agents still might experience bad results, as their distance can be linear in the number of participating agents. Cycle replacement fixes this shortcoming by trading in some individual efficiency.

Theorem 3.4. *Cycle replacement achieves a best case and worst case social distance of* $\mathrm{Dist}(o) = n + n_c - 2c$ *and thus achieves the social optimum distance. Simultaneously it guarantees a worst case distance of* 2 *for each individual agent.*

Proof. In each star the algorithm generates from a cycle C in G_s, the center and exactly one of the other agents are directly adjacent to their preferred

partners. All other agents of the original cycle have a distance of 2 to their preferred partner. This yields a total cost of $2 \cdot |C| - 2$ for each cycle C, matching the social optimum cost in Lemma 3.1. All other agents are directly adjacent to their preferred partner, as the rest of the algorithm is unchanged from Cycle breaking. Thus, the Cycle replacement achieves the social optimum of Theorem 3.1 and additionally no agent has a distance greater than 2 to her preferred partner. □

Thus, by replacing cycles with stars, the mechanism is *efficient*, as it only produces *efficient* networks. The created networks are *individually efficient* for all agents who are not part of a cycle, and for two agents of each cycle in G_s. For all other agents it is *2-approximately individually efficient*, and thereby establishes fairness amongst all agents, without compromising global efficiency.

3.2.2. Individual Rationality

It is unlikely that agents would participate in a mechanism, that brings them losses. Depending on the parameter β and which of the three social choice functions is used, networks might or might not be individually rational:

Theorem 3.5. *Iterative attachment does not insure individual rationality. Cycle breaking guarantees this property for all but one agent per cycle C with $|C| - 1 > \beta$. Cycle replacement produces individual rational networks for all agents exactly then if either G_s does not contain a cycle C with $|C| > 2$ or $\beta \geq 2$; by adding fair payments to Cycle breaking, it achieves the same.*

Lemmas 3.2 to 3.4 prove this theorem. We start with the first social choice function, that might harm agents, as it is only individually rational for some outcomes:

Lemma 3.2. *Building trees with Iterative attachment does not ensure non-negative utility for agents. Some networks generated by this algorithm are individually rational.*

Proof. As agents can only derive constant profit β from the result, this lemma follows directly from Theorem 3.2 and the corresponding proof. □

The second social choice function does better, as it produces a good network for most agents, however still not for all of them:

Lemma 3.3. *Cycle breaking guarantees non-negative utility for all but one agent per cycle C with $|C| - 1 > \beta \geq 1$.*

Proof. As agents can only derive constant profit β from the result, this lemma follows directly from Theorem 3.3 and the corresponding proof. □

The mechanism could consider charging a fee ($p_i = -1$) from all but two agents in a cycle C with $|C| > 2$, that have not been separated from their preferred partners, and paying that money to the agent who has been selected in Line 4 ($p_i = |C| - 3$). By doing so it remains *budget-balanced*, all results concerning *efficiency* would still apply as before, but in terms of rationality the mechanism would achieve the same as by using Cycle replacement. By replacing cycles with stars, this third social choice function does worse for some, and better for most values of β than the previous two:

Lemma 3.4. *Cycle replacement produces individually rational networks if and only if either the strategy graph G_s does not contain cycles with size greater two, or $\beta \geq 2$.*

The same does Cycle breaking if augmented with payments as described above.

Proof. The first part follows from Theorem 3.4 and the corresponding proof.

For the second part observe, that the utility for all agents outside cycles C with $|C| > 2$ does not change. The same holds for two agents inside such cycles, who are adjacent to their partners. All other agents in those cycles have a new utility of $u_{i,1} = \beta - 2$, as they are either adjacent to their respective friend but have to pay $p_i = 1$ to the mechanism, or are in distance $|C| - 1$ but receive a payment of $|C| - 3$ from the mechanism. These are exactly the utilities of the corresponding result of Cycle replacement. □

Note that while it might seem disappointing that none of the three social choice functions is *individual rational* for all inputs and all values of β, this is in fact not possible to achieve for the given utility function:

Theorem 3.6. *For $1 \leq \beta < 2$ there cannot be any social choice function for trees that achieves individual rationality for strategy graphs G_s with any cycle C with $|C| > 2$.*

Proof. Assume the contrary and inspect a strategy graph G_s that includes such a cycle. Then, in the output graph o all agents have to be directly adjacent to their preferred partners. (For any agent a_i who is not, $d_o(a_i, t_i) \geq 2$ and thus $u_{i,1}(o) < 0$.) This however means that o includes a cycle of length ≥ 3. □

3.2.3. Stability

Considering swap stability, one can easily see that only in rare cases no agent can improve her utility:

Theorem 3.7. *Only if G_s does not contain cycles of length greater 2 can a resulting tree be swap-stable. Cycle breaking and Cycle replacement then always produce such a result, Iterative attachment does so only in some cases.*

Proof. If G_s does not contain cycles with more than two nodes, then there exists a swap-stable tree o, in which all agents are adjacent to their respective preferred partner, thus no agent can improve her utility. Both Cycle breaking and Cycle replacement construct that tree. Iterative attachment can for some inputs build the same graph, as Theorem 3.2 shows, but can also yield unstable results as illustrated in Figure 3.3.

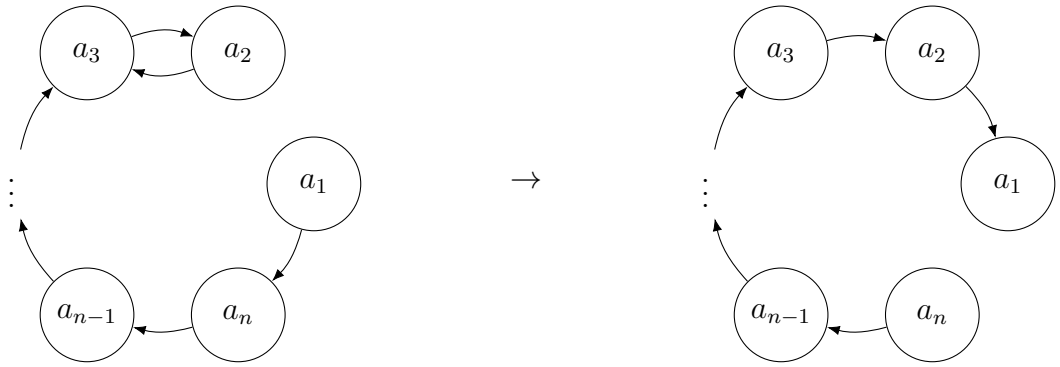


Figure 3.3.: Iterative attachment fails to produce a *swap-stable* tree from the feasible input on the left: In the resulting tree on the right, a_1 could improve her utility by dropping the edge from a_2 and building one to a_n.

If G_s however contains a cycle with length greater 2, there must be an agent in such a longer cycle that is not adjacent to her preferred partner in o, as the resulting network is a tree. Thus, such an agent can always improve her utility by dropping the edge through which she currently reaches her preferred partner, and establishing a direct edge to this partner instead. After that, the network would again be substance to such a swap, and the process would cycle indefinitely. □

Iterative attachment does not perform any better concerning anarchy stability, whereas the other two social choice functions always produce *anarchy stable* trees:

Theorem 3.8. *Trees produced by Iterative attachment not always fulfill anarchy stability. Both Cycle breaking and Cycle replacement are anarchy stable.*

Proof. Figure 3.3 shows, that Iterative attachment does not necessarily produce outputs that are *anarchy-stable*, as independent from the order in which agents are allowed to build their preferred edges, in an anarchic process all agents would be adjacent to their preferred partners. Thus, in the depicted example, a_1 would be better off without the mechanism. As described in the proof of Theorem 3.2, the algorithm can for some inputs also produce networks in which all agents are adjacent to their preferred partners; in these cases the output is *anarchy-stable*.

Cycle breaking produces only *anarchy-stable* trees, as any agent a_i, that is not adjacent to her preferred partner t_i in a resulting tree is part of a cycle C in G_s. If there are $|C|$ agents in that cycle, she has a distance of $|C| - 1$ to t_i. This is exactly as far away as if she was last in the permutation of agents in the anarchic network design process.

When using Cycle replacement, all agents are either adjacent to their respective preferred partner, or are part of a cycle in G_s and have a distance of 2 to their partners. As before, the worst case instance puts such an agent last in the permutation and would thus lead to a distance depending on the number of agents in the corresponding cycle in G_s. Therefore, Cycle replacement guarantees *anarchy stability* and on top ensures, that no agent is placed more than two hops from her preferred partner. □

Note that the end of the proof highlights a property of the output trees of Cycle replacement, that is stronger than anarchy stability:

Corollary. *Cycle replacement not only produces anarchy-stable trees, but for some agents even guarantees that they are strictly closer to their preferred partners with the mechanism than without.*

This result is a strong argument for this social choice function, if agents decide whether to participate in a mechanism or not. If they risk to end up with linear distance to their preferred partner, they might prefer an anarchic process over a mechanism-based network design. Should however such a mechanism guarantee constant distances, the willingness to both participate in the process and accept its result is likely to be much higher.

3.2.4. Truthfulness

So far we always assumed, that agents are honest about their type and declare it as their strategy, that is $s = t$. However, as agents are selfish, if an agent

a_i can increase her utility by declaring some $s_i \neq t_i$, she would do so. Conveniently, all social choice functions presented so far encourage agents to be honest:

Theorem 3.9. *No agent a_i with the utility function $u_{i,1}$ can solely strictly improve her utility by declaring some strategy $s_i \neq t_i$.*

Proof of Theorem 3.9 for Iterative attachment. As the utility function $u_{i,1}$ is strictly monotonically decreasing, improving utility is equivalent to narrowing the distance to the respective preferred partner. Thus, only those agents not adjacent to their preferred partners can possibly strictly improve their utility.

Let a_i be an agent who can strictly improve her utility. In networks generated by Iterative attachment, a_i has entered the graph before her preferred partner $a_j = t_i = s_i$ which in turn prefers some third agent $a_k \neq a_i$. As we inspect a worst-case instance in which Iterative attachment does not choose uniformly at random but choses agents according to their natural order, $i < j$ holds.

Note that all edges in the generated output graph o are either edges from G_s or connect the next "random" agent to the existing graph; in the later case they connect the first unconnected agent to a_1.

If $i = 1$, the agent's strategy s_i is never even considered in the algorithm, thus declaring some $s_i \neq t_i$ does not influence the output graph. For $i > 1$ the agent a_i is placed adjacent to a_1. We distinguish two cases based on the location of t_i in the output graph o:

- If $t_i = a_j$ is adjacent to a_1, then either she prefers agent $a_1 = s_j$, or the preferred partner of t_i enters the graph after t_i and a_1 is selected as edge endpoint due to the fixed order of agents in our worst-case instance.

  Both ways, a_i cannot achieve to be placed closer than distance 2, as neither of the conditions for edges described above can be fulfilled by altering s_i.

- If t_i is not adjacent to a_1, then she is connected to a_1 via a path that consists only of agents who entered the graph after a_i and possibly a_i, as shown in Figure 3.4: All trees in $o \setminus \{a_1\}$ consist only of edges from G_s and thus agents therein join the graph consecutively, as the iterative attachment process prefers to attach agents whose preferred partners are already in the graph. Therefore, if one of the agents in such a tree entered o before a_i, the whole tree, including t_i would. As t_i is known to enter the graph after a_i, there is no agent that a_i could declare as strategy $s_i \neq t_i$ who would place her closer to t_i than her connection to a_1.

□

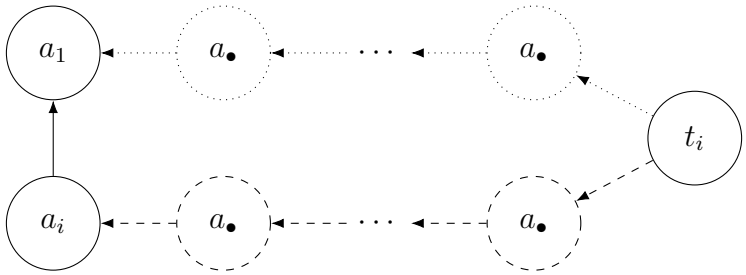


Figure 3.4.: If t_i is not adjacent to a_1, she is connected to a_1 via a path that may (dashed lines) or may not (dotted lines) include a_i. Either way, all agents in these paths except for a_1 and a_i entered the graph after a_i and all $a_\bullet$ have higher indices than a_i.

Proof of Theorem 3.9 for Cycle breaking. All agents who are not part of a cycle of length > 2 in G_s are already adjacent to their preferred partners and cannot improve. The same holds for all but one agent in each cycle. Let this agent be a_i, then (a_i, t_i) is the edge the social choice function chose to remove from the graph to break the cycle. Thus, in the result network, the agent is connected to her preferred partner via her predecessors in the cycle, meaning the shortest path from a_i to t_i runs opposite to the edge directions through the cycle to t_i, as shown in Figure 3.5. Due to our worst-case rule for random selection, i is the smallest index in the whole cycle; particularly the index of a_i is smaller than the indices of all other agents on that shortest path p.

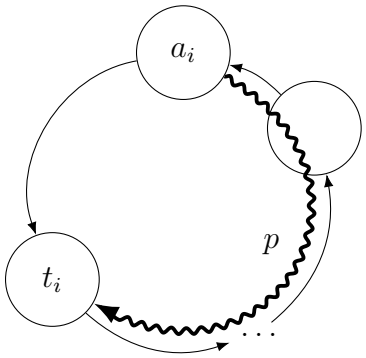


Figure 3.5.: As i is the smallest index in the initial cycle, the mechanism will never remove an edge on a_i's initial shortest path p (thick) to t_i, leaving her distance to t_i unchanged no matter which s_i she declares.

By choosing $s_i \neq t_i$, agent a_i still is part of a cycle. If that cycle does not include any agent of the original cycle apart from a_i, she is connected to t_i with the unique path p an does not improve her distance. Should the new

cycle overlap with the original one, it still holds that a_i has a lower index than other agents on p, and the mechanism will break the cycle along one of the edges not part of p. Thus, the mechanism won't ever remove an edge on p, and a_i cannot narrow the distance to her preferred partner by lying. □

Proof of Theorem 3.9 for Cycle replacement. As in the previous proof, only agents in cycles of length > 2 in G_s are relevant. All agents but two in such a cycle have a distance of 2 to their respective preferred partners; the two remaining being the center of the constructed star and the agent whose preferred partner is that center. Both are adjacent to their preferred partners and cannot improve.

Let a_i be one of the agents with distance 2 to her preferred partner. If she declares some $s_i \neq t_i$, she would still be part of a cycle in the new strategy graph. Should this new cycle include t_i, then it also includes all other agents of the original cycle, and the mechanism would again chose neither a_i nor t_i as center for the cycle. The same happens if the new cycle does not include t_i, but her preferred partner (which is her second neighbor in the cycle apart from a_i). In case the new cycle does not include either of those two, then t_i is not directly adjacent to the center of the new constructed star. Either way, the distance between a_i and her preferred partner is ≥ 2. □

3.3. Distance Bounded Profit

A second utility function assumes, that agents make a constant profit of $\beta \in \mathbb{R}, \beta > 1$ from the connection to their preferred partners, if they are within a constant maximum distance, similar to the *bounded maximum-distance* game proposed by Bilò et al. [11]. In contrast to their game however, an agent is not punished if the distance between her and her friend is too large, but she "only" cannot derive any value from the network.

We choose 2 as maximum distance, as by Theorem 3.4 no agent would have any incentive to accept a larger distance:

$$u_{i,2}(o) = v_{i,2}(o) = \begin{cases} \beta & d_o(a_i, t_i) \leq 2 \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

Again, the function limits the maximum distance an agent finds acceptable, but instead of having to minimize the distance, this time agents only have an incentive to be "close enough" to their preferred partner. As $u_{i,2}(o) \geq 0$ for all outcomes o, the mechanism is *individually rational*, independent of the used

social choice function. Similarly all statements on distances in Section 3.2.4 still hold, making the mechanism again *truthful* for any input strategy vector.

As before, we will at first assume that agents play honestly and declare their true type.

3.3.1. Social Welfare and Efficiency

For Iterative attachment, the efficiency results from Section 3.2.1 still apply, as the social choice function does not guarantee that an agent comes closer than $\Omega(n)$ to her preferred partner. Similarly, Cycle breaking again produces *individually efficient* networks for all but one agent per large cycle. However, the algorithm now fails to achieve *efficiency* for all networks with a cycle with more than three agents. Cycle replacement now produces both *efficient* and *individually efficient* networks for any input.

Theorem 3.10. *A network created by Cycle breaking is efficient if and only if there is no cycle of size > 3 in G_s. The social choice function produces individually efficient networks for all agents except one in such a larger cycle.*

The tree mechanism with Cycle replacement as social choice function is efficient and individually efficient.

Proof. If there is no cycle C with $|C| > 3$ in G_s, Cycle breaking creates a tree in which all agents are in distance ≤ 2 to their respective preferred partner, thus all derive β from interacting with their partner. In each cycle consisting of more than three agents, there is exactly one agent (the one chosen in Line 4) with a distance of at least 3, who does not derive any value.

With Cycle replacement, by Theorem 3.4, all agents are within distance 2 of their preferred neighbors and thus gain the maximum value β. □

3.3.2. Stability

Concerning anarchy stability, Theorem 3.8 also holds for the new utility function $u_{i,2}$. The results for swap stability remain unchanged for Iterative attachment, but improve for the other two:

Theorem 3.11. *Iterative attachment only produces swap-stable networks in some cases. Cycle breaking creates swap-stable networks for all input strategy graphs G_s that do not include a cycle of more than three agents. For arbitrary inputs, Cycle replacement always builds swap-stable networks.*

Proof. The argument for the Iterative attachment is the same as for Theorem 3.7.

If G_s does not contain cycles with more than three agents, Cycle breaking constructs an output tree in which all agents are within distance 2 to their respective preferred partner, therefore no agent can improve her utility by swapping an edge. For larger cycles, the agent selected in Line 4 has a larger distance and thus can improve her utility by dropping the edge to her predecessor in the cycle and establishing a direct connection to her preferred partner instead.

By Theorem 3.10, all agents receive their maximum utility in any network created by Cycle replacement and therefore no agent can improve. □

Note that in contrast to Theorem 3.7, the swapping process in case of Cycle breaking function does not necessarily cycle indefinitely, as the agents instead can end up building a star such as Cycle replacement does.

3.4. Additional Profit from Close Neighbors

There would hardly be any point in having a connected network, if agents only were to derive value from their preferred partners. Therefore we extend the utility function by taking into account all agents connected to a_i. From each other agent a_j that she decides to interact with, she derives some constant value — $\beta \in \mathbb{R}, \beta > 1$ from her preferred partner like before, and $\gamma \in \mathbb{R}, \gamma > 1$ from all others — whilst having to pay $d_o(a_i, a_j)$ for using the edges on the path to this agent.

As we expect agents to be rational and selfish, we assume they only decide to interact with some a_j if this interaction strictly increases utility:

$$\begin{aligned} u_{i,3}(o) &= v_{i,3}(o) \\ &= \max\left\{\left(\beta - d_o(a_i, t_i)\right), 0\right\} + \sum_{a_j \notin \{a_i, t_i\}} \max\left\{\left(\gamma - d_o(a_i, a_j)\right), 0\right\} \end{aligned} \tag{3.3}$$

Now, β and γ effectively limit the maximum distance that a_i finds acceptable for interacting with her friend and the remaining agents. We call the set of non-preferred agents that yield positive utility for a_i the γ-neighborhood of a_i. Usually $\beta \geq \gamma$ should hold (as the preferred partner ought to bring at least as much profit for the agent than some other partner). For $\beta > \gamma$, being close to the preferred partner could possibly even compensate for few other nearby agents and vice versa.

With the same argument as for $u_{i,2}$, all resulting trees and mechanisms are *individually rational*, as $u_{i,3}$ cannot drop below zero.

We now consider a small selection of values for both β and γ and analyze networks according to their swap-stability.

3.4.1. All neighbors yield identical profit

For $\beta = \gamma$, an agent's only goal is to be able to reach as many agents a_j as possible with a distance of $d_o(a_i, a_j) < \gamma$. This is again very similar to the bounded-distance model of Bilò et al. [11].

If $\beta = \gamma \leq 2$ holds, agents are only interested in the number of direct neighbors. As dropping an edge to one neighbor and establishing a connection to some other does not increase this number, all networks are *swap-stable* for such β and γ.

In case $\beta = \gamma > 2$, agents are also interested in neighbors farther away. This leads to a narrow set of *swap-stable* output networks.

Theorem 3.12. *For $\beta = \gamma > 2$ in the utility function $u_{i,3}$, a tree network is swap-stable if and only if it is a star.*

We prove this in two steps: First we establish that no other networks can be swap-stable in Lemma 3.5, then we show that all stars are stable in Lemma 3.6.

Lemma 3.5. *If, for the utility function $u_{i,3}$, we choose $\beta = \gamma > 2$, no swap-stable tree can contain a path consisting of more than two edges. Therefore all swap-stable trees must be stars in this case.*

Proof. Assume the contrary and inspect a path of length ≥ 3 with agents $a_1, \ldots, a_4$ and their (potentially empty) subtrees $T_1, \ldots T_4$, as shown in Figure 3.6.

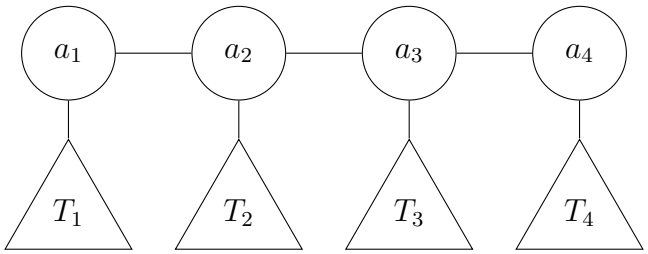


Figure 3.6.: Either a_1 or a_4 can improve her utility by swapping an edge to a_3 or a_2, respectively.

Let $u_{i,3}(T_k)$ be the utility agent i derives from subtree T_k if she is directly adjacent to a_k. Note that for $\beta = \gamma$ the utility does not depend on the individual agent, thus we can drop the i and write $u_3(T_k)$.

If $u_3(T_2) = u_3(T_3)$, agent a_1 can strictly improve her utility by dropping the edge to a_2 and establishing a direct connection to a_3, as the sum of utility she derives from a_2, a_3, T_1, T_2, T_3 remains unchanged, and she increases her

utility derived from a_4 and T_4. For $u_3(T_2) < u_3(T_3)$, she additionally increases the utility sum derived from a_2, a_3, T_2, T_3. The third case, $u_3(T_2) > u_3(T_3)$, is symmetric to the second; this time a_4 can strictly increase her utility by swapping the edge to a_3 with an edge to a_2.

Therefore, any path of length greater 2 contains an agent who can strictly improve her utility by swapping an edge. Thus, the only possible *swap-stable* trees for $\beta = \gamma > 4$ are stars. □

While Lemma 3.5 only discards all non-stars as potential *swap-stable* trees, we can also show, that all trees indeed fulfill the properties for swap stability:

Lemma 3.6. *If, for the utility function $u_{i,3}$, we choose $\beta = \gamma > 2$, all stars are swap-stable.*

Proof. There have to be at least three agents for any swap to be possible, thus any star with $n < 3$ is *swap-stable*.

As the central agent of a star is already connected to all other agents, she cannot swap an edge. If any of the other agents swaps an edge, she would still be adjacent to exactly one agent whilst changing the size of her 2-neighborhood from $n-2$ to 1 and her 3-neighborhood from 0 to $n-3$. For $n > 3$ this strictly decreases her utility, for $n = 3$ her neighborhoods and therefore also her utility remain unchanged. □

Note that the only assumptions the proofs of Lemmas 3.5 and 3.6 make, are that the utility function does not distinguish preferred partners and others, that the 2-neighborhood yields positive utility, and that the utility function is monotonic in the sense that agents farther away yield less utility than closer ones. The results therefore hold not only for $u_{i,3}$, but for all utility functions that fulfill those assumptions and Theorem 3.13 follows:

Theorem 3.13. *For any utility function that*

- *does not distinguish between preferred and non-preferred agents*
- *yields positive utility for the 2-neighborhood of an agent*
- *is monotonic, thus yields more profit for closer agents than for distant ones*

the set of swap-stable trees is exactly the set of stars.

As discussed in Section 1.1.1, stars are rather unpractical for large networks. We hold the assumptions to be reasonable, except for the equal treatment of all nearby agents. Therefore, Theorem 3.13 is a strong advocate for our position: It is unrealistic to assume agents would value all other players as equally important partners.

None of the presented social choice functions always produces a star as output network: The first two only construct a star, if the initial graph G_s almost is a star; Cycle replacement also transforms an input cycle into a star. Except for those rare cases, all three functions generate output trees that are not *swap-stable* for $\beta = \gamma > 2$.

3.4.2. Preferred partners yield more profit

As preferred partners ought to be of greater importance to an agent, than other players, it is reasonable to assume, that $\beta > \gamma$. In this case, the utility function no longer becomes oblivious of the preferred partner; an agent now might achieve higher utility with a smaller neighborhood if the preferred partner is among those close neighbors. For small γ that (in addition to her preferred partner) only enable an agent to derive profit from the directly adjacent non-preferred agents, Cycle replacement is *swap-stable*, as it exactly matches the stability criterion:

Theorem 3.14. *If, for the utility function $u_{i,3}$, we choose $\beta > \gamma$ and $\gamma \leq 2$, a network o is stable if and only if there is no agent a_i with $d_o(a_i, t_i) > 2$.*

Proof. For $\gamma \leq 2$, agents can again not change the size of their γ-neighborhood by swapping one edge. Therefore only if an agent a_i is not adjacent to her preferred partner t_i she can possibly make a profitable swap. If the input strategy graph G_s does not contain a cycle of more than two agents, there exist output trees in which all agents are adjacent to their respective preferred partner (and both Cycle breaking and Cycle replacement produce such a tree).

For graphs with larger cycles there must be some agent, that is not adjacent to her preferred partner in the output tree. She then can possibly strictly increase her utility by dropping the first edge on the path to her preferred partner and building a direct edge instead. (Building a direct edge is guaranteed to be more profitable, than to only narrow the distance.) Thus, in a *swap-stable* network, the following equation must hold for each such agent and

the respective distance $d_p > 1$ to her preferred partner:

$$\overbrace{(\beta - 1)}^{\text{preferred, gained}} - \overbrace{(\beta - d_p)}^{\text{preferred, lost}} - \overbrace{(\gamma - 1)}^{\text{non-preferred, lost}} \leq 0 \tag{3.4}$$

$$d_p \leq \gamma \tag{3.5}$$

As $\gamma \leq 2$, networks are stable if and only if no agent is in distance > 2 to her preferred partner. □

For larger γ it is yet unknown how stable trees might look like and how they could be generated.

3.5. Exponential Profit Decrease

For our fourth and last utility function for simple tree mechanisms, we build on $u_{i,3}$. This time however, agents not only have to pay for edge usage, but generated value decreases exponentially with increasing distance:

$$\begin{aligned} u_{i,4}(o) = v_{i,4}(o) = &\max\left\{\left(\beta \cdot 2^{1-d_o(a_i,t_i)} - d_o(a_i,t_i)\right), 0\right\} \\ &+ \sum_{a_j \notin \{a_i,t_i\}} \max\left\{\left(\gamma \cdot 2^{1-d_o(a_i,a_j)} - d_o(a_i,a_j)\right), 0\right\} \end{aligned} \tag{3.6}$$

Like before, a_i only interacts with friends close enough to still yield value — again determined by β and the γ-neighborhood — which makes all resulting trees and mechanisms *individually rational*, as $u_{i,4}$ cannot drop below zero.

For $\beta = \gamma > 4$, Theorem 3.13 applies, as $u_{i,4}$ fulfills the same necessary properties as $u_{i,3}$. Thus, again the set of stable networks is the set of stars for such parameters. However, for small values of γ and $\beta > \gamma$, no tree can be *swap-stable* if there is a cycle with more than two agents in the strategy graph G_s.

Theorem 3.15. *If, for the utility function $u_{i,4}$, we choose $\beta > \gamma$ and $\gamma \leq 4$, either no network tree is swap-stable or the strategies do not form a cycle with more than two agents in G_s.*

Proof. Except for minor adjustments, the proof is almost identical to Theorem 3.14:

For $\gamma \leq 4$, agents can not change the size of their γ-neighborhood by swapping one edge, hence again the special role of strategy graphs G_s without large cycles.

For graphs with larger cycles there must be some agent, that is not adjacent to her preferred partner in the output tree. She can possibly strictly increase her utility in the same way as in Theorem 3.14, and the following equation must hold for each such agent in a *swap-stable* network and the respective distance $d_p > 1$ to her preferred partner:

$$\left(\beta \cdot 2^{1-1} - 1\right) - \left(\beta \cdot 2^{1-d_p} - d_p\right) - \left(\gamma \cdot 2^{1-1} - 1\right) \leq 0 \tag{3.7}$$

$$\beta - 1 - \frac{\beta}{2^{d_p-1}} + d_p - \gamma + 1 \leq 0 \tag{3.8}$$

$$\beta \cdot \left(1 - \frac{1}{2^{d_p-1}}\right) \leq \gamma - d_p \tag{3.9}$$

$$\beta \leq (\gamma - d_p) \cdot \frac{2^{d_p}}{2^{d_p} - 2} \tag{3.10}$$

At the same time, $\beta > \gamma$ must hold, thus we get

$$\gamma < (\gamma - d_p) \cdot \frac{2^{d_p}}{2^{d_p} - 2} \tag{3.11}$$

$$2^{d_p} \cdot \gamma - 2 \cdot \gamma < 2^{d_p} \cdot \gamma - 2^{d_p} \cdot d_p \tag{3.12}$$

$$\gamma > 2^{d_p-1} \cdot d_p \tag{3.13}$$

As d_p is at least 2, we get $\gamma > 4$ which contradicts $\gamma \leq 4$. In summary, if there is an agent, that is not adjacent to her preferred partner, it is always beneficial for her to swap an edge. □

Again, for larger values of γ, no results on *swap-stability* in trees are known.

4. Weighted Strategies

So far, agents were only allowed to declare a single friend to the mechanism. Now we instead expect them to rank all $n-1$ other agents in a tuple in decreasing order of importance to them. Therefore G_s is a complete and weighted directed graph.

We assume, that an agent a_i derives the value $\beta \cdot 2^{1-r}$, $\beta \in \mathbb{R}, \beta > 1$ from a reachable agent $t_i[r]$ that is ranked on r-th position in t_i. She again has to pay the hop distance $d_o(a_i, t_i[r])$ to that agent for using the edges of graph o, and will thus only interact with agents that are close or important enough to yield positive utility:

$$u_{i,5}(o) = v_{i,5}(o) = \sum_{r=1}^{n-1} \max\left\{\left(\frac{\beta}{2^{r-1}} - d_o(a_i, t_i[r])\right), 0\right\} \tag{4.1}$$

4.1. A Greedy Social Choice Function

It would appear logical to build a social choice function on top of G_s, that selects an optimal subgraph satisfying some specified property. As mentioned in Section 1.1.2, this often leads to NP-complete problems, leaving no hope for constructing a generic poly-time computable mechanism[4]. Instead, we will use a greedy heuristic that ensures such a property and favors — compliant with the agents' strategies — highly ranked edges (with small edge weight) over ones with low priority (and high edge weight).

For that, the social choice function is given a predicate P that is initially fulfilled for G_s. The function then iterates over all edges in decreasing edge weight order, removing each edge from the graph, given that the removal does not lead to violation of P. The function presented in Algorithm 4.1 can be implemented to run in $\mathcal{O}(n^2 \cdot f_P(n))$ time, where f_P describes the complexity of the predicate P. It is thus poly-time computable for those P that can be tested in polynomial time.

4 However, we suggest to work on optimizing for specific properties in our Future Work section.

Algorithm 4.1: Reverse edge deletion

Input: Agents $A = \{a_1, \ldots, a_n\}$, Strategy vector $s = (s_1, \ldots, s_n)$, Predicate P
Output: Generated Graph G with $P(G)$ is TRUE

```
G ← G_s;
for r = n − 1 to 1 do
    for i = 1 to n do
        if P(G \ {(a_i, s_i[r])}) then
            G ← G \ {(a_i, s_i[r])};
return G
```

4.2. Creating Weighted Trees

The predicate in the reverse edge deletion algorithm is especially capable of enforcing connectivity constraints. We will again restrict ourselves to trees, as they are easier to analyze than highly connected resulting networks.

By selecting $P(G) \equiv$ (G is connected), the social choice function returns a minimum spanning tree; the resulting algorithm is essentially CONSTRUCTION A' by Kruskal [38], an algorithm he introduced as dual to the more popular minimum spanning tree algorithm named after him.

For this social choice function and the utility function $u_{i,5}$, we show two negative results; namely that results are not necessarily efficient and that — in contrast to all social choice functions in Chapter 3 — agents have an incentive not to play honestly.

4.2.1. Efficiency

Even though a minimum spanning tree optimizes the total weight of the resulting network, this is not the same as maximizing social welfare:

Theorem 4.1. *For $u_{i,5}$, any minimum spanning tree is efficient if $\beta \leq 2$. This is not always true for larger values of β.*

Proof. For $1 < \beta \leq 2$, only first-ranked agents that are directly adjacent yield profit. As a minimum spanning tree maximizes the number of edges of weight 1, the result is efficient.

If $\beta > 2$, additionally second-ranked adjacent agents yield profit. Consider $2 < \beta \leq 4$ and an intermediate state of G during the deletion of edges of Algorithm 4.1, shown in Figure 4.1. Assume, only edges of weight 1 remain,

and T_1 and T_2 are already proper MSTs for the respective agents therein. Furthermore, apart from the drawn edges, no agent from T_1 ranked an agent of T_2 higher than 2, and vice-versa. Thus, the only agents that can profit from one of the crossing edges, are the respective endpoints of those edges. The algorithm would now remove edge (a_1, a_2) and return the resulting minimum spanning tree.

However, if (a_4, a_3) was removed instead, the resulting MST would have a higher social welfare: Let $u_5(T_1)$ and $u_5(T_2)$ be the sum of the utility of all agents without both crossing edges. Then, adding (a_1, a_2) would add profit to both endpoints and the total utility would be $u_5(T_1)+u_5(T_2)+(\beta-1)+(\frac{\beta}{2}-1)$, as both a_1 and a_2 benefit from the connection. Using edge (a_4, a_3) instead would only accumulate to $u_5(T_1) + u_5(T_2) + (\beta - 1)$ total utility, as due to $\beta \leq 4$ a third-ranked a_4 cannot yield profit for a_3.

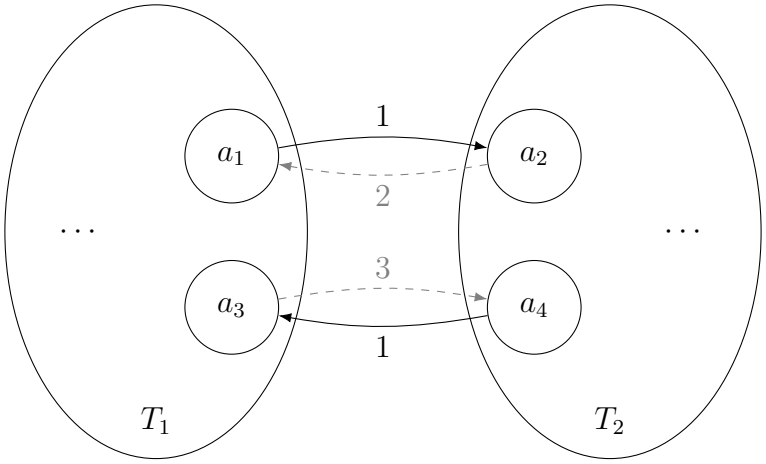


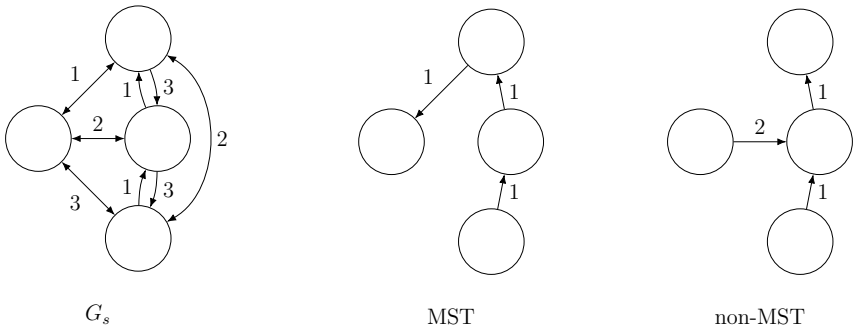
Figure 4.1.: T_1, T_2 are already MST with only edges of weight 1; The dashed edges have been removed in previous iterations of Algorithm 4.1. The two possible MST do not yield the same profit.

□

Therefore, a minimum spanning tree does no longer guarantee efficiency for $\beta > 2$. Theorem 4.2 establishes an even stronger result, that shows that in some cases, not only are some MSTs inefficient, but even all of them.

Theorem 4.2. *For $u_{i,5}$ and $\beta > 2$, an output network with higher total edge weight can be more efficient than a minimum spanning tree.*

Proof. As Figure 4.2 shows, trees with higher total edge weight can be more efficient than any MST. □



G_s MST non-MST

Figure 4.2.: For $\beta = 12$, and the strategy graph (left), the unique minimum spanning tree (center) yields a social welfare of 64. The optimal solution (right) achieves the higher value of 66, as the distance costs are lower. As always, the direction of an edge shows which player declared it as part of her strategy.

4.2.2. Truthfulness

With the added complexity of ranked preferences, the agents also gained the ability to lie about their true type, in order to increase their utility.

Theorem 4.3. *If the utility an agent derives from another player decreases with distance, the mechanism is not truthful.*

Proof. Consider the strategy graph and the resulting networks in Figure 4.3. If a_1 plays truthfully (top), the second-ranked edge between a_2 and a_4 is preferred by the algorithm.

Should she however wrongly claim to have a_3 as her first-ranked agent, the result is a minimum spanning tree in which she decreased the distance to a_3 without any changes to her other hop distances. Any utility function that yields more value for closer agents and yields positive profit for third-ranked partners would therefore encourage her to lie about her true type.

□

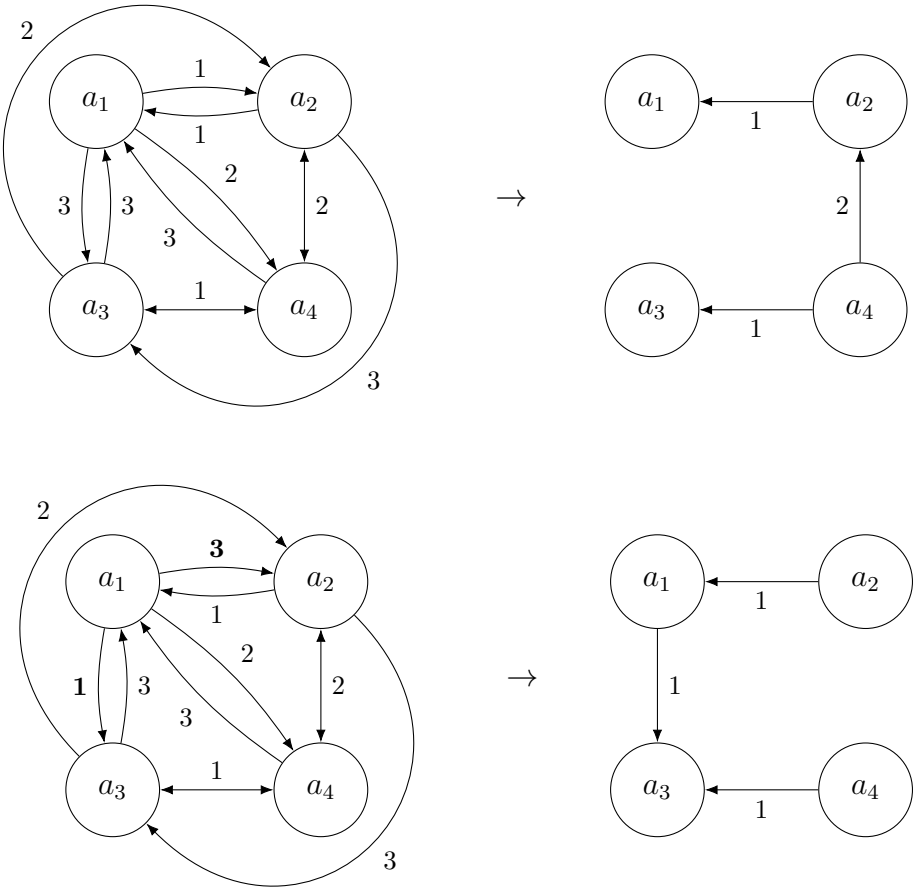


Figure 4.3.: Left are strategy graphs, right resulting minimum spanning trees. Agent a_1 can alter her true strategy (top) and swap her first- and third-ranked agent (bottom) to decrease her distance to a_3 and therefore increase her utility.

5. Bounded degrees and budgets

Our last model basically combines ideas of Laoutaris et al. [39] and Ehsani et al. [24]: Actors now no longer can establish and maintain any number of connections to neighbors, but are only able to deal with up to k neighbors. In contrast to Laoutaris et al. [39], this bound applies both to outgoing and incoming edges, thus we limit the total degree of nodes, like Avin et al. [7] do in their network design model. Real-world examples for such degree limits occur in technology, where the number of ports a component provides, is limited, or also in social interaction networks, where the time a person can spend on communication is fixed.

Both type and strategy of agents is shaped the same way, as we assume that agents have a bounded budget to finance edges: Each agent a_i has $\lfloor \frac{k}{2} \rfloor$ (unweighted) preferred partners and thus declares a subset $s_i \subseteq (A \backslash \{a_i\}), |s_i| = \lfloor \frac{k}{2} \rfloor$ of other agents as strategy. This way, at least half of an agents' "ports" is available for incoming connections.

Consider a set of agents, in which all favor one agent a^* as neighbor. Clearly, even if no other constraints except for the degree bound apply, $\leq k^d$ agents can be within distance d of agent a^*. Therefore agents have to accept a worst-case distance of $\Theta(\log n)$ to their preferred partner.

5.1. Building Snowflakes

As the initial strategy graph might violate the degree constraints, the main task of the social choice function is to establish compliance with the given bounds. A simple, centralized network design approach could thus construct a full $(k-1)$-ary tree to ensure both connectedness and that no agent is farther than $\Theta(\log n)$ from any of her preferred partners. However, in our opinion, a good output network more closely resembles the strategy graph G_s. For that, Algorithm 5.1 moves edges from nodes with too high degree away to the closest node with potential for additional neighbors.

Whereas the algorithm does not guarantee logarithmic distance between

every agent and her preferred partners, it only adjusts G_s locally and thereby avoids to change parts of the strategy graph that are fully compliant with the degree constraint. Figure 5.1 shows an example for that strategy, and also motivates the algorithm's name.

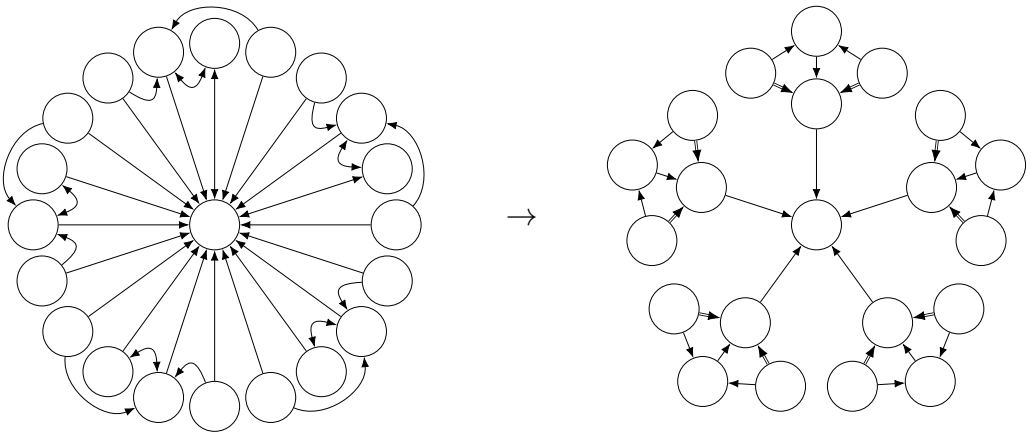


Figure 5.1.: Many agents who favor the same partners are spread out into a snowflake-like shape. In this case, $k = 5$, thus every agent declared two preferred partners. Double drawn arrows indicate edges added in Line 11 of Algorithm 5.1.

Given an overfull node a with more than k neighbors, consider the graph rooted at the node a, and all connected nodes arranged in layers according to their hop distance to a. The algorithm now progresses in increasing distance d to a and one by one removes the direct edge of an adjacent agent a_i with $a \in s_i$ until the degree constraint is met.

It does so in two different ways: Either, there is such an a_i that — after (a_i, a) is cut — still has distance d to a, then this operation is performed and, visually speaking, a_i falls down to layer d (see Figure 5.2). Or, there exists an agent a_j with distance $d - 1$ to a whose degree allows for an additional edge. Then the mechanism selects an agent a_i with $a \in s_i$ from those agents, who are farthest away from a after removal of their direct edge to a. Again, (a_i, a) is cut, but this time replaced with an edge (a_i, a_j), again resulting in a_i now being in layer d (see Figure 5.3). The distance d is increased whenever neither of those two actions is possible, to iteratively fill all layers with the maximum number of agents.

If Algorithm 5.1 terminates, G is obviously a graph without any vertices that violate the degree bound. Theorem 5.1 establishes, that the algorithm indeed always terminates and thus proves the correctness of Algorithm 5.1.

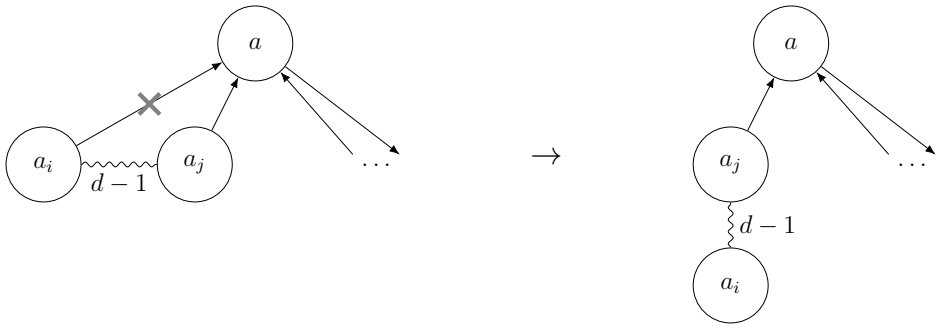


Figure 5.2.: A direct edge between a_i and a is cut, if a_i afterwards still has a path of length d to a.

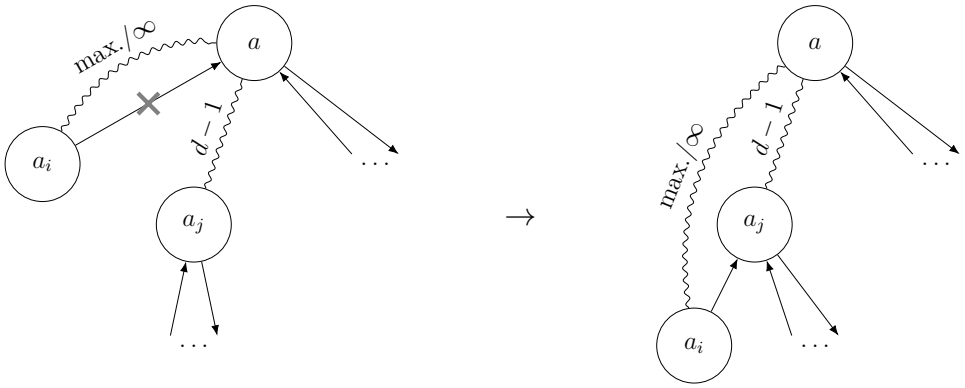


Figure 5.3.: Agent a_i has maximum distance (or no connection) to a, after its direct edge is cut. The direct edge is replaced with an edge to a_j, a node with degree $< k$ and distance $d-1$ to a instead.

Theorem 5.1. *The social choice function in Algorithm 5.1 always terminates and generates a graph in which all nodes have at most k neighbors.*

Proof. Observe that if the loop in Line 5 always terminates, the whole function does: The algorithm never increases the degree of any node above k, and each time the loop in Line 5 exits, the number of nodes with more than k neighbors is decreased by one.

In order to prove, that the inner loop always terminates, we assume the contrary. As both the if-block and the else if-block strictly decrease the degree of a by one, this can only happen if neither condition applies and the else block is executed in each iteration. This especially implies that the loop was executed for all values $2 \leq d \leq n$.

Now check, that the first condition effectively checks for a cycle of length $d+1$ from a over some $a_i \in N$, traveling over $d-2$ agents in $A \setminus N$, over some

Algorithm 5.1: Snowflake creation

Input: Agents $A = \{a_1, \ldots, a_n\}$, Strategy vector $s = (s_1, \ldots, s_n)$
Output: Generated Graph

$G \leftarrow G_s \setminus \{(a_i, a_j) \mid i > j \wedge (a_j, a_i) \in G\}$;
while $\exists a \in A$ *with* $\deg(a) > k$ **do**
  $N \leftarrow \{a_i \in A \mid a \in s_i\}$;
  $d \leftarrow 2$;
  while $\deg(a) > k$ **do**
    if $\exists a_i \in N$ *with* $d_{G \setminus \{(a_i,a)\}}(a_i, a) = d$ **then**
      $G \leftarrow G \setminus \{(a_i, a)\}$;
      $N \leftarrow N \setminus \{a_i\}$;
    else if $\exists a_j \in A$ *with* $d_G(a_j, a) = d - 1 \wedge \deg(a_j) < k$ **then**
      $a_i \leftarrow \operatorname{argmax}_{a_i \in A}\{d_{G \setminus \{(a_i,a)\}}(a_i, a) \mid a_i \notin \text{path of } d_G(a_j, a)\}$;
      $G \leftarrow (G \setminus \{(a_i, a)\}) \cup \{(a_i, a_j)\}$;
      $N \leftarrow N \setminus \{a_i\}$;
    else
      $d \leftarrow d + 1$;
return G

a_j that is adjacent to a back to the beginning of the cycle. Such a cycle is depicted on the left side of Figure 5.2. As the condition is not fulfilled anymore for any $2 \leq d \leq n$, none of the agents from N is part of any cycle that includes the root a. Therefore, no two agents from N are part of the same component in $G \setminus \{a\}$. As $\deg(a) > k$, there are $|N| > \frac{k}{2}$ such components, which all (in G) have one edge to a, and none from a into that component.

Now, inspect the size and calculate the average node degree of any of those components; let x be the number of agents within it. In total, the out-degree of all x agents is at most $x \cdot \lfloor \frac{k}{2} \rfloor$. As one of those edges points to a, the component contains at most $x \cdot \lfloor \frac{k}{2} \rfloor \cdot 2 - 1$ edge endpoints, which leads to an average node degree $< k$. For that some agent in that component has to have less than k neighbors. As the distance of this agent to a is between 1 and $n-1$, the else-if condition would have applied.

Therefore, d cannot grow above n, which contradicts the assumption that the inner loop does not terminate. It directly follows from the first observation and the nature of the loop conditions, that the whole algorithm terminates and the result graph G fulfills the node degree bound k. □

5.2. Anarchy is bad

The main point of this mechanism is to show, that with certain constraints on the network infrastructure, an anarchic process can yield much worse results than a mechanism can provide. For our model with a maximum node degree k, consider a set A^* of $k+1$ agents, whose preferences form a clique. Assume, these players are important for many others, who as well list some agents of A^* as preferred partners. Should however, in an anarchic process, the clique form, before the other players have a chance to connect to agents therein, those others will end up completely disconnected from agents in A^*, as all nodes in the clique already have k neighbors.

The Snowflake creation on the other hand retains connectivity of components of G_s, and only adjusts distances in order to meet the degree constraint. Thus, all agents are guaranteed to be connected to their preferred partners when participating in the mechanism.

Corollary. *Snowflake creation is anarchy stable for all utility functions that punish an agent for not being connected to all her preferred partners. Moreover, it not only matches the quality of an anarchic process, but performs better by guaranteeing connectivity between each agent a_i and her respective preferred partners s_i.*

5.3. Possible Extensions

In contrast to the tree mechanisms, we no longer have a fixed number of edges in the final graph. Therefore, it is reasonable to assume, that agents now would have to pay for their edges.

Snowflake creation ensures that all degree bounds are met, but it does not attempt to cater requests for other properties like connectedness. If agents were to profit more from closer friends (like in $u_{i,1}$), the *budget balanced* social choice function could be extended to charge agents close to their preferred partners some payment and use it to finance additional edges between otherwise disjoint components.

6. Conclusion and Future Work

This work explored the possibilities of mechanisms for network creation in the context of several different utility functions and with respect to preferable properties of resulting networks. When generating trees, for which agents are allowed to declare a single preferred partner, our Cycle replacement algorithm proved to be a social choice function that fulfills all properties for a bounded distance utility model. The Snowflake creation algorithm highlighted, that (in contrast to an uncoordinated network creation) mechanisms are able to establish important properties, such as the mere chance for all agents to at least be connected to their preferred partners.

We have shown, that for richer player strategies and more complex utility function, our algorithms often are not yet able to guarantee stability. Apart from finding answers to the not yet analyzed fields of Table 1.1, this leaves a set of intriguing mechanism design challenges open for future work. As we only started to experiment with the model, there is also a lot of potential for interesting modifications of it.

Average-case analysis of expected outcomes So far, we have only looked into the most important properties of our mechanisms and the resulting networks. In particular, our analysis always considered worst-case instances. However, agents might be more interested in the expected outcome in the average case, or even for specific realistic distributions of the input preferences. Future work should establish a way to analyze the models accordingly. This also includes the implementation of the proposed algorithms and a statistical analysis of their results.

Characterization of Equilibria By challenging our output networks to the same stability considerations as in classical network creation games, the same problems arise: How do stable networks look like, do they always exist, and how might they be created? This is especially interesting in the context of complex utility functions like $u_{i,3}$ and networks, that are at least biconnected.

Mechanism Design All mechanisms presented in this work are built to fulfill some specific property, such as connectedness or individual efficiency, and only do this in the context of simple graph structures. The more general problem of finding (approximately) optimal subgraphs of a strategy graph remains open and interesting for a multitude of properties. We do for example believe, that the snowflake creation presented in Section 5.1 can be improved to guarantee logarithmic worst-case distances between all agents and their respective preferred partners. More extensions are already discussed in Section 5.3.

Edge ownership and payments In our model section we introduced the common concept, that edges can be owned and payed for by agents. So far, our examples do not make use of this. Theorem 3.5 already gives a hint, how payments might help to produce fairness amongst agents. Combined with the concept of network stability, agents could be charged little for edges planned by the mechanism, and more for edges they want to selfishly establish on top of the outcome. The additional budget of the mechanism could then be used to finance edges for global properties such as low diameter.

Network robustness While relatively easy to analyze, trees often are not desirable for real-world networks, as they are vulnerable to edge- or node-failures. In order to create more stable networks, we would for example like to incorporate the model of Goyal et al. [30] with attack and immunization in our mechanisms. Could maybe a social choice function be capable of creating networks and efficiently select immunized nodes?

Scale-free graphs and evolving networks In many real-world networks, one can find similar distinct properties, such as a power-law distribution on node degrees, or high clustering. Generating networks with mechanisms, that implement those properties, might help to understand how real-world graph structures form. Additionally, large networks usually are not formed once and then kept unchanged, but evolve as agents enter or exit the network, and preferences are updated. Thus, our model for mechanisms should be extended to cope with ongoing changes to the input data.

(Im)possibility theorems Finally, it would be beneficial to establish clear boundaries of what mechanisms can or cannot achieve. Similar to the impossibility theorems for voting systems [6, 28, 51], we would like to know, whether preferable properties of networks can be achieved simultaneously, or whether there are impossible combinations.

Appendix

A. Game Theory Examples

A.1. Prisoners' Dilemma

"Two men, charged with a joint violation of law, are held separately by the police." So begins the story Albert Tucker invented in 1950 to present a psychological study of RAND researchers Merrill Flood and Melvin Dresher to a broader audience [49, p. 116 ff.]. Those two prisoners are suspected to have committed a major crime, but the police does not have enough evidence to convict them for more than a lesser charge. They offer both prisoners the same deal: Either, he testifies against the other, or refuses to take the deal and stays silent. Should only one prisoner testify, he is free to go, whereas his partner is convicted. In case neither of them talks, both do time for the lesser charge. However, if both testify against the respective other, both can be sentenced for committing the crime. Table A.1 shows the full payoff matrix with the sentences in each case.

	B refuses deal	**B testifies**
A refuses deal	1 year, 1 year	3 years, 0 years
A testifies	0 years, 3 years	2 years, 2 years

Table A.1.: The payoff matrix of the Prisoners' Dilemma game. Cells show *Profit of A*, *Profit of B*.

Assuming, the prisoners are selfish and are only interested in minimizing their own time in jail, both will testify: Independent of whether the other player takes the deal or not, a prisoner is guaranteed to serve one year less by selling out the other criminal. It follows, that the game only is in Nash equilibrium if both players take the deal.

The social optimum, however, would be, to cooperate and serve one year each. Intriguingly, this would also reduce the prison term for both individually.

By being rational in the game, the players therefore produce the worst social outcome of four years totally served and simultaneously harm themselves.

Nisan et al. [46, chap. 1.1.2] extend the Prisoners' Dilemma and show a version that is played by n agents; in their case n countries deciding about whether or not to invest in pollution control. Just like with the prisoners in the original game, their extension shows that the only stable state is for all countries to not control pollution. In the end, each country has to bear costs that are linear in the number of players, instead of constant costs if all had decided to not pollute.

A.2. Hawk-Dove Game

In the scenario of the Hawk-Dove game, two animals compete for a piece of food. Should both choose the aggressive *Hawk* strategy, they would destroy the food, and get nothing. If they both decide for the passive *Dove* strategy, they evenly share the food. Would however one play *Hawk*, and one *Dove*, the more aggressive animal gets most of the food, leaving only little for the passive one. Table A.2 shows the payoffs of the game as presented by Easley and Kleinberg [23, chap. 6.6].

	B plays *Dove*	**B plays *Hawk***
A plays *Dove*	3, 3	1, 5
A plays *Hawk*	5, 1	0, 0

Table A.2.: The payoff-matrix of the Hawk-Dove game played with 6 total "units" of food. Cells show *Profit of A*, *Profit of B*.

In contrast to the Prisoners' Dilemma, the social worst case is not a stable state in this game. However, as there are two Nash equilibria — the two cases in which there is one hawk and one dove — the behavior of the players is impossible to predict: Both have an interest to maximize their food share, so they would like to play aggressive, but only if the other does not. The game therefore has an inherent risk of ending up in the (both global and individual) worst case that destroys the very resource the players competed for. This result is quite important for numerous real-world situations: Confrontations between companies or countries can be modeled as Hawk-Dove game.

A.3. Tragedy of the Commons

While the more recent and more popular publication by Hardin [33] coined its title, the story itself is attributed to Lloyd [43]: A group of herdsman collectively owns some land, on which their animals graze. Each herdsman receives the full proceeds from his cattle. If the land can provide enough food for all animals, his revenue only depends on the size of his own herd.

The tragedy takes its course, when herdsmen consider whether to add animals to their herd or not: Additional revenue goes to the individual, additional grazing affects the whole community. As long as the effect of overgrazing on the single herdsman is smaller than the additional proceeds, herds will grow larger and larger — up to the point where overgrazing ruins the land.

Again, rational individual decisions can lead to a worst-case outcome for individuals and the whole society. Nisan et al. [46, chap. 1.1.2] show the same effect in a scenario about bandwidth-usage. They run exemplary numbers for their version and conclude, that there is an unstable state of the game, with which the n players were to get about $\frac{n}{4}$-times more bandwidth than in the stable state.

A.4. Braess's Paradox

This example of Braess's paradox [13] presented here is taken from Steimle [53]:

Given the left graph shown in Figure A.1, n agents are trying to route traffic (e.g. data or actual road traffic) from s to t. For using an edge e_i, an agent has to bear costs of c_i (in money, latency, ...), given as edge weights in the graph. Connections e_2 and e_3 are large enough to cope with any amount of traffic and yield constant costs, whereas the cost of e_1 and e_4 depends on the number of agents using those edges.

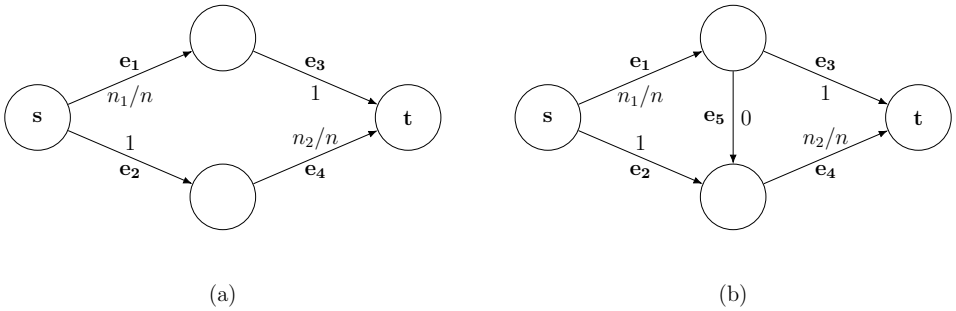


Figure A.1.: The original (a) and augmented (b) network of Braess's paradox. Traffic flows from s to t with n_i being the number of agents using edge e_i and n the total number of agents.

The optimal distribution would now be for $\frac{n}{2}$ agents to pick the route (e_1, e_3), and the other half to go for (e_2, e_4), resulting in costs of 1.5 for each agent. Thereby, both all individual costs and the overall routing cost would be minimized. This distribution is stable in the sense that no agent can pay less by deviating from her chosen route; it is therefore a Nash equilibrium. As in this case the optimum and the equilibrium are identical, the agents lose nothing due to lack of coordination.

However, consider a new edge e_5, free to use for anyone as shown on the right of Figure A.1. In the augmented network all agents would choose (e_1, e_5, e_4) instead of their previous route, as $c_1 \leq c_2$ and $c_4 \leq c_3$. The optimal flow, in which both the overall costs and all individual costs are minimized, remains unchanged, but the only stable flow (thus, the only equilibrium in the routing game on the augmented graph) is for everyone to choose (e_1, e_5, e_4). But by doing so, the cost for each agent increases to 2 leaving the agents worse off than without the additional edge.

For Figure A.1, a central authority could either control the traffic flow or simply remove e_5 from the network to restore the social optimum. Thus, using the sum of individual costs as social cost, the Price of Anarchy is $\frac{\text{cost of worst equilibrium}}{\text{cost of social optimum}} = \frac{4}{3}$ in the example.[5]

5 Note that [53] confuses values and claims the Price of Anarchy was $\frac{3}{4}$.

References

[1] S. Albers. On the Value of Coordination in Network Design. *SIAM Journal on Computing*, 38(6):2273–2302, Jan. 2009. ISSN 0097-5397, 1095-7111. doi: 10.1137/070701376. (Cited on page 5.)

[2] S. Albers, S. Eilts, E. Even-Dar, Y. Mansour, and L. Roditty. On Nash Equilibria for a Network Creation Game. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 89–98, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics. ISBN 978-0-89871-605-4. (Cited on pages 5 and 6.)

[3] N. Alon, E. D. Demaine, M. Hajiaghayi, and T. Leighton. Basic Network Creation Games. In *Proceedings of the Twenty-Second Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '10, pages 106–113, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0079-7. doi: 10.1145/1810479.1810502. (Cited on pages 5 and 18.)

[4] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The Price of Stability for Network Design with Fair Cost Allocation. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '04, pages 295–304, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 978-0-7695-2228-9. doi: 10.1109/FOCS.2004.68. (Cited on page 5.)

[5] E. Anshelevich, A. Dasgupta, É. Tardos, and T. Wexler. Near-Optimal Network Design with Selfish Agents. *Theory of Computing*, 4:77–109, Aug. 2008. doi: 10.4086/toc.2008.v004a004. (Cited on page 5.)

[6] K. J. Arrow. A Difficulty in the Concept of Social Welfare. *Journal of Political Economy*, 58(4):328–346, 1950. ISSN 0022-3808. (Cited on pages 8 and 52.)

[7] C. Avin, K. Mondal, and S. Schmid. Demand-Aware Network Designs of Bounded Degree. *arXiv:1705.06024 [cs]*, May 2017. (Cited on pages 5 and 46.)

[8] V. Bala and S. Goyal. A Noncooperative Model of Network Formation. *Econometrica*, 68(5):1181–1229, 2000. ISSN 0012-9682. (Cited on page 5.)

[9] N. Berger, M. Feldman, O. Neiman, and M. Rosenthal. Dynamic Inefficiency: Anarchy without Stability. In *Algorithmic Game Theory*, Lecture Notes in Computer Science, pages 57–68. Springer, Berlin, Heidelberg, Oct. 2011. ISBN 978-3-642-24828-3 978-3-642-24829-0. doi: 10.1007/978-3-642-24829-0_7. (Cited on page 6.)

[10] D. Bilò and P. Lenzner. On the Tree Conjecture for the Network Creation Game. In R. Niedermeier and B. Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science (STACS 2018)*, volume 96 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-062-0. doi: 10.4230/LIPIcs.STACS.2018.14. (Cited on page 7.)

[11] D. Bilò, L. Gualà, and G. Proietti. Bounded-Distance Network Creation Games. In *Internet and Network Economics*, Lecture Notes in Computer Science, pages 72–85. Springer, Berlin, Heidelberg, Dec. 2012. ISBN 978-3-642-35310-9 978-3-642-35311-6. doi: 10.1007/978-3-642-35311-6_6. (Cited on pages 5, 33, and 36.)

[12] L. Blume, D. Easley, J. Kleinberg, R. Kleinberg, and É. Tardos. Network Formation in the Presence of Contagious Risk. *ACM Trans. Econ. Comput.*, 1(2):6:1–6:20, May 2013. ISSN 2167-8375. doi: 10.1145/2465769.2465771. (Cited on page 5.)

[13] D. Braess. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung Operations Research - Recherche Opérationnelle*, 12 (1):258–268, Dec. 1968. ISSN 0340-9422, 1432-5217. doi: 10.1007/BF01918335. (Cited on pages 6 and 55.)

[14] A. Chauhan, P. Lenzner, A. Melnichenko, and M. Münn. On Selfish Creation of Robust Networks. In *Algorithmic Game Theory*, Lecture Notes in Computer Science, pages 141–152. Springer, Berlin, Heidelberg, Sept. 2016. ISBN 978-3-662-53353-6 978-3-662-53354-3. doi: 10.1007/978-3-662-53354-3_12. (Cited on page 5.)

[15] A. Chauhan, P. Lenzner, A. Melnichenko, and L. Molitor. Selfish Network Creation with Non-uniform Edge Cost. In *Algorithmic Game Theory*, Lecture Notes in Computer Science, pages 160–172. Springer,

Cham, Sept. 2017. ISBN 978-3-319-66699-0 978-3-319-66700-3. doi: 10.1007/978-3-319-66700-3_13. (Cited on page 5.)

[16] H.-L. Chen and T. Roughgarden. Network Design with Weighted Players. *Theory of Computing Systems*, 45(2):302, Aug. 2009. ISSN 1432-4350, 1433-0490. doi: 10.1007/s00224-008-9128-8. (Cited on page 5.)

[17] H.-L. Chen, T. Roughgarden, and G. Valiant. Designing Networks with Good Equilibria. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 854–863, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics. (Cited on page 5.)

[18] J. Corbo and D. Parkes. The Price of Selfish Behavior in Bilateral Network Formation. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Principles of Distributed Computing*, PODC '05, pages 99–107, New York, NY, USA, 2005. ACM. ISBN 978-1-58113-994-5. doi: 10.1145/1073814.1073833. (Cited on page 5.)

[19] A. Cord-Landwehr, M. Hüllmann, P. Kling, and A. Setzer. Basic Network Creation Games with Communication Interests. In *Algorithmic Game Theory*, Lecture Notes in Computer Science, pages 72–83. Springer, Berlin, Heidelberg, 2012. ISBN 978-3-642-33995-0 978-3-642-33996-7. doi: 10.1007/978-3-642-33996-7_7. (Cited on page 5.)

[20] E. D. Demaine, M. Hajiaghayi, H. Mahini, and M. Zadimoghaddam. The Price of Anarchy in Network Creation Games. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing*, PODC '07, pages 292–298, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-616-5. doi: 10.1145/1281100.1281142. (Cited on page 5.)

[21] E. D. Demaine, M. Hajiaghayi, H. Mahini, and M. Zadimoghaddam. The Price of Anarchy in Cooperative Network Creation Games. In S. Albers and J.-Y. Marion, editors, *26th International Symposium on Theoretical Aspects of Computer Science*, volume 3 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 301–312, Dagstuhl, Germany, 2009. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-939897-09-5. doi: 10.4230/LIPIcs.STACS.2009.1839. (Cited on page 5.)

[22] Dionne R. and Florian M. Exact and approximate algorithms for optimal network design. *Networks*, 9(1):37–59, Oct. 2006. ISSN 0028-3045. doi: 10.1002/net.3230090104. (Cited on page 3.)

[23] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press, Cambridge, 2010. Draft version: June 10, 2010. (Cited on page 54.)

[24] S. Ehsani, S. S. Fadaee, M. Fazli, A. Mehrabian, S. S. Sadeghabad, M. Safari, and M. Saghafian. A Bounded Budget Network Creation Game. *ACM Trans. Algorithms*, 11(4):34:1–34:25, Apr. 2015. ISSN 1549-6325. doi: 10.1145/2701615. (Cited on pages 5 and 46.)

[25] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. On a Network Creation Game. In *Proceedings of the Twenty-Second Annual Symposium on Principles of Distributed Computing*, PODC '03, pages 347–351, New York, NY, USA, 2003. ACM. ISBN 978-1-58113-708-8. doi: 10.1145/872035.872088. (Cited on pages ii, iii, 4, 5, 6, 9, and 16.)

[26] D. Gale and L. S. Shapley. College Admissions and the Stability of Marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962. ISSN 0002-9890. doi: 10.2307/2312726. (Cited on page 8.)

[27] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990. ISBN 978-0-7167-1045-5. (Cited on page 3.)

[28] A. Gibbard. Manipulation of Voting Schemes: A General Result. *Econometrica*, 41(4):587–601, 1973. ISSN 0012-9682. doi: 10.2307/1914083. (Cited on pages 8 and 52.)

[29] L. Gourvès and J. Monnot. Three Selfish Spanning Tree Games. In *Internet and Network Economics*, Lecture Notes in Computer Science, pages 465–476. Springer, Berlin, Heidelberg, Dec. 2008. ISBN 978-3-540-92184-4 978-3-540-92185-1. doi: 10.1007/978-3-540-92185-1_52. (Cited on page 5.)

[30] S. Goyal, S. Jabbari, M. Kearns, S. Khanna, and J. Morgenstern. Strategic Network Formation with Attack and Immunization. In *Web and Internet Economics*, Lecture Notes in Computer Science, pages 429–443. Springer, Berlin, Heidelberg, Dec. 2016. ISBN 978-3-662-54109-8 978-3-662-54110-4. doi: 10.1007/978-3-662-54110-4_30. (Cited on pages 5 and 52.)

[31] R. Graham, L. Hamilton, A. Levavi, and P.-S. Loh. Anarchy Is Free in Network Creation. In *Algorithms and Models for the Web Graph*, Lecture Notes in Computer Science, pages 220–231. Springer, Cham,

Dec. 2013. ISBN 978-3-319-03535-2 978-3-319-03536-9. doi: 10.1007/978-3-319-03536-9_17. (Cited on page 5.)

[32] Y. Halevi and Y. Mansour. A Network Creation Game with Nonuniform Interests. In *Internet and Network Economics*, Lecture Notes in Computer Science, pages 287–292. Springer, Berlin, Heidelberg, Dec. 2007. ISBN 978-3-540-77104-3 978-3-540-77105-0. doi: 10.1007/978-3-540-77105-0_28. (Cited on page 5.)

[33] G. Hardin. The tragedy of the commons. The population problem has no technical solution; it requires a fundamental extension in morality. *Science (New York, N.Y.)*, 162(3859):1243–1248, Dec. 1968. ISSN 0036-8075. (Cited on page 55.)

[34] M. Hoefer. Non-Cooperative Tree Creation. *Algorithmica*, 53(1):104–131, Jan. 2009. ISSN 0178-4617, 1432-0541. doi: 10.1007/s00453-007-9014-9. (Cited on page 5.)

[35] L. Hurwicz. Optimality and informational efficiency in resource allocation processes. In K. J. Arrow, S. Karlin, and P. Suppes, editors, *Mathematical Methods in the Social Sciences: 1959 ; Proceedings of the First Stanford Symposium on Mathematical Methods in the Social Sciences*, number 4 in Stanford mathematical studies in the social sciences, pages 27–47. Stanford University Press, Stanford, California, 1960. ISBN 978-0-8047-0021-4. (Cited on page 8.)

[36] D. S. Johnson, J. K. Lenstra, and A. H. G. R. Kan. The complexity of the network design problem. *Networks*, 8(4):279–285, Dec. 1978. ISSN 1097-0037. doi: 10.1002/net.3230080402. (Cited on page 3.)

[37] E. Koutsoupias and C. Papadimitriou. Worst-Case Equilibria. In *STACS 99*, Lecture Notes in Computer Science, pages 404–413. Springer, Berlin, Heidelberg, Mar. 1999. ISBN 978-3-540-65691-3 978-3-540-49116-3. doi: 10.1007/3-540-49116-3_38. (Cited on page 6.)

[38] J. B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956. ISSN 0002-9939. doi: 10.2307/2033241. (Cited on pages 2 and 42.)

[39] N. Laoutaris, R. Rajaraman, R. Sundaram, and S.-H. Teng. A bounded-degree network formation game. *arXiv:cs/0701071*, Jan. 2007. (Cited on pages 5 and 46.)

[40] N. Laoutaris, L. J. Poplawski, R. Rajaraman, R. Sundaram, and S.-H. Teng. Bounded Budget Connection (BBC) Games or How to Make Friends and Influence People, on a Budget. In *Proceedings of the Twenty-Seventh ACM Symposium on Principles of Distributed Computing*, PODC '08, pages 165–174, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-989-0. doi: 10.1145/1400751.1400774. (Cited on page 5.)

[41] P. Lenzner. On Dynamics in Basic Network Creation Games. In *Algorithmic Game Theory*, Lecture Notes in Computer Science, pages 254–265. Springer, Berlin, Heidelberg, Oct. 2011. ISBN 978-3-642-24828-3 978-3-642-24829-0. doi: 10.1007/978-3-642-24829-0_23. (Cited on page 5.)

[42] P. Lenzner. Greedy Selfish Network Creation. In *Internet and Network Economics*, Lecture Notes in Computer Science, pages 142–155. Springer, Berlin, Heidelberg, Dec. 2012. ISBN 978-3-642-35310-9 978-3-642-35311-6. doi: 10.1007/978-3-642-35311-6_11. (Cited on page 5.)

[43] W. F. Lloyd. *Two Lectures on the Checks to Population*. 1833. (Cited on page 55.)

[44] M. Mihalák and J. C. Schlegel. The Price of Anarchy in Network Creation Games is (Mostly) Constant. In *Proceedings of the Third International Conference on Algorithmic Game Theory*, SAGT'10, pages 276–287, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 978-3-642-16169-8. (Cited on pages 5 and 7.)

[45] N. Nisan and A. Ronen. Algorithmic Mechanism Design (Extended Abstract). In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, STOC '99, pages 129–140, New York, NY, USA, 1999. ACM. ISBN 978-1-58113-067-6. doi: 10.1145/301250.301287. (Cited on page 9.)

[46] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, USA, 2007. ISBN 978-0-521-87282-9. (Cited on pages 7, 8, 54, and 55.)

[47] Nobel Media AB. The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 2007. https://www.nobelprize.org/nobel_prizes/economic-sciences/laureates/2007/, 2007. (Cited on page 8.)

[48] E. Ostrom. *Governing the Commons: The Evolution of Institutions for Collective Action*. The Political economy of institutions and decisions.

Cambridge University Press, Cambridge ; New York, 1990. ISBN 978-0-521-37101-8 978-0-521-40599-7. (Cited on page 8.)

[49] W. Poundstone. *Prisoner's Dilemma: John Von Neumann, Game Theory and the Puzzle of the Bomb*. Doubleday, New York, NY, USA, 1st edition, 1992. ISBN 978-0-385-41567-5. (Cited on page 53.)

[50] R. C. Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, Nov. 1957. ISSN 0005-8580. doi: 10.1002/j.1538-7305.1957.tb01515.x. (Cited on page 2.)

[51] M. A. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10(2):187–217, Apr. 1975. ISSN 0022-0531. doi: 10.1016/0022-0531(75)90050-2. (Cited on pages 8 and 52.)

[52] A. J. Scott. The optimal network problem: Some computational procedures. *Transportation Research*, 3(2):201–210, July 1969. ISSN 0041-1647. doi: 10.1016/0041-1647(69)90152-X. (Cited on page 3.)

[53] J. Steimle. *Algorithmic Mechanism Design: Eine Einführung*. Informatik im Fokus. Springer-Verlag, Berlin Heidelberg, 2008. ISBN 978-3-540-76401-4. (Cited on pages 55 and 56.)

[54] W. Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *The Journal of Finance*, 16(1):8–37, 1961. ISSN 0022-1082. doi: 10.2307/2977633. (Cited on page 8.)

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst habe. Alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken wurden als solche kenntlich gemacht.

Potsdam, 30.05.2018

Stefan Neubert